UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE CIENCIAS MATEMÁTICAS

UNIDAD DE POSTGRADO

Combinación de clasificadores en redes neurales

TESIS

para optar el Grado Académico de Magíster en Estadística

AUTOR

Cleto De la Torre Dueñas

ASESOR

Carlo Veliz C.

Lima – Perú

2007

DEDICATORIA

A mis queridos padres Gualberto (✞) y Carmela.

A mi esposa Yeny Maritza.

A mi hijo Diego Alberto.

A mis hermanos: Hernán, Flavio (✞), Vilma, Nery Rus y Leida.

AGRADECIMIENTO

Quiero expresar en primer lugar mi más sincero agradecimiento a mi asesor Mgt. Carlos Veliz C. por su gran apoyo, predisposición y dedicación en el desarrollo de la presente tesis, sin los que no hubiera sido posible la misma.

También deseo expresar mi agradecimiento a todos los docentes de la Unidad de Post- Grado de la FACULTAD DE CIENCIAS MATEMATICAS de la UNMSM por sus enseñanzas.

Finalmente mi agradecimiento a la UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO, institución en la cual trabajo, por el apoyo que me brindó para realizar estudios de maestría en Estadística.

RESUMEN

COMBINACIÓN DE CLASIFICADORES EN REDES NEURALES

En este trabajo se describe la red neuronal como modelo estadístico no lineal, y se presenta aplicaciones de los métodos de combinación de clasificadores ‘’bagging’’ y ‘’boosting’’ en redes neuronales a las bases de datos sonar e iris, como una alternativa de reducción de la tasa de mala clasificación del método de redes neuronales.

Palabras Claves

-Redes neuronales.

-Combinación de clasificadores.

-Bagging.

-Boosting.

SUMMARY

COMBINATION OF CLASSIFIERS IN NEURAL NETWORKS

In this thesis is described a neural networks as statistic model non linear and it study the method of the combination of classifiers bagging and boosting in neural networks as an alternative of reduction of the wrong rate's classifiers for the method of neural networks. As application of this procedures are analyzed the base of dates. They are very known as '' sonar'' and ''iris''.

Key Words:

- Neural networks.

-Combination of classifiers.

-Bagging.

-Boosting.

ÍNDICE GENERAL

CAPÍTULO I: **REDES NEURONALES**

CAPITULO II: **COMBINACIÓN DE CLASIFICADORES. MÉTODOS: BAGGING Y BOOSTING.**

CAPITULO III: **APLICACIÓN.**

CONCLUSIONES Y RECOMENDACIONES

PLANTEAMIENTO.

Es difícil imaginar el estudio de datos provenientes de las distintas áreas del conocimiento humano que no involucren problemáticas de clasificación, regresión, etc. El tratamiento de estos problemas se ha facilitado con el surgimiento de computadoras de amplia capacidad y con una serie de métodos estadísticos y de inteligencia artificial como el método de **Redes Neurales, llamados también Redes neuronales.**

Una red neuronal artificial es un modelo que trata de imitar la neurofisiología del cerebro humano. A partir de datos de una muestra representativa, la red aprende a encontrar tendencias o patrones en los datos. Específicamente una red neuronal, es un modelo que involucra, de una manera irrestricta y dinámica a modelos de regresión no lineal, modelos discriminantes, y modelos de reducción de datos. Las redes neuronales pueden ayudar a hacer predicciones acerca de problemas del mundo real.

En general, las redes neuronales son especialmente útiles para problemas analíticos que:

- Están relacionados con un volumen muy grande de datos, en donde es difícil especificar un modelo parámetrico.
- Están contenidos en datos que muestran una no-linealidad significativamente difícil de caracterizar.
- Están contenidos en datos, en los que existen patrones estables muy sutiles o profundamente escondidos.
- Requieren el uso iterativo de los datos para detectar patrones o tendencias.

Las Redes Neuronales surgen como una alternativa de estudio en ‘’problemas de clasificación’’, ante las limitaciones de los modelos clásicos, rígidos y con características de tipo lineal.

Sin embargo, estas técnicas son ineficientes, como por ejemplo, en problemas de clasificación en donde existen categorías con muy pocos elementos. En tales casos los métodos ’’ Boosting y Bagging”, proporcionan cierta mejora a la solución de los problemas indicados. La idea subyacente en estas técnicas es la ‘’ combinación‘’ de clasificadores que sin aumentar el sesgo reduce la variabilidad de los resultados.

OBJETIVOS.

OBJETIVO GENERAL

Analizar el efecto sobre el error de clasificación en redes neuronales cuando se combinan los clasificadores, Boosting y Bagging.

OBJETIVOS ESPECIFICOS

- Estudiar el método de redes neuronales desde el punto de vista estadístico.
- Estudiar y comparar los métodos Bagging y Boosting, en términos de error de clasificación, aplicados a las redes neuronales.
- Comparar la eficiencia de los métodos Bagging y Boosting en base a la red neuronal.

HIPÓTESIS

Las técnicas de combinación de clasificadores: Bagging y Boosting en redes neuronales, reducen la tasa de mala clasificación.

CAPÍTULO I: REDES NEURONALES.

1.1 Introducción.

La teoría de Redes Neuronales está inspirada en los conocimientos existentes del funcionamiento del sistema nervioso de los seres vivos. A partir de estos estudios, surgen modelos con un conjunto de propiedades específicas, como:

- La habilidad de adaptarse al entorno, aprendiendo a proporcionar la respuesta adecuada ante los estímulos que reciba de este entorno. Este aprendizaje se plasma en la modificación de los llamados ‘’pesos’’ de las conexiones entre los distintos elementos, llamados ‘’neuronas’’ que forman la red.
- Generalización y organización de la información

Una Red Neuronal (RNA) imita en cierto modo la estructura física y el modo de funcionamiento del cerebro humano. El conocimiento es adquirido por la red a través de un proceso que se denomina **APRENDIZAJE**, emulando el proceso de aprendizaje de las neuronas interconectadas en el cerebro. El aprendizaje en el sentido estadístico corresponde a la estimación de los pesos de las conexiones de la red neuronal.

1.2 **Antecedentes**

Las primeras ideas fueron introducidas por McCulloch y Pits (1943). Posteriormente, Rosenblatt (1958,1962) introdujo el concepto de ‘’perceptron’’ y lo trató de aplicar a la clasificación. Pero no fue hasta 1985, con los trabajos de Hinton, Rumellhart y Williams quienes presentaron los algoritmos de

"Backpropagation", que éstas comenzaron a llamar la atención de la comunidad científica. [2]

En Estadística, Ripley (1993), Chen y Titterigton (1994) fueron pioneros en la incursión de los estadísticos en las redes neuronales. [7]

Actualmente existen muchos trabajos que se realizan y publican cada año; sin embargo gran parte de las aplicaciones están relacionadas con la inteligencia artificial.

1.3 Elementos de una Red Neuronal.

Los elementos de una red neuronal (ver figura 1.1) son los siguientes:

Neuronas.- Llamados también elementos de proceso. Las neuronas se agrupan en capas, las que reciben la información a procesar y forman la capa de entrada, esta información se transmite a ciertos elementos internos (neuronas) que se ocupan de su procesamiento, estas neuronas forman las capas ocultas. Una vez que ha finalizado el procesamiento, la información llega a las unidades (neuronas) de salida formando estas la capa de salida.

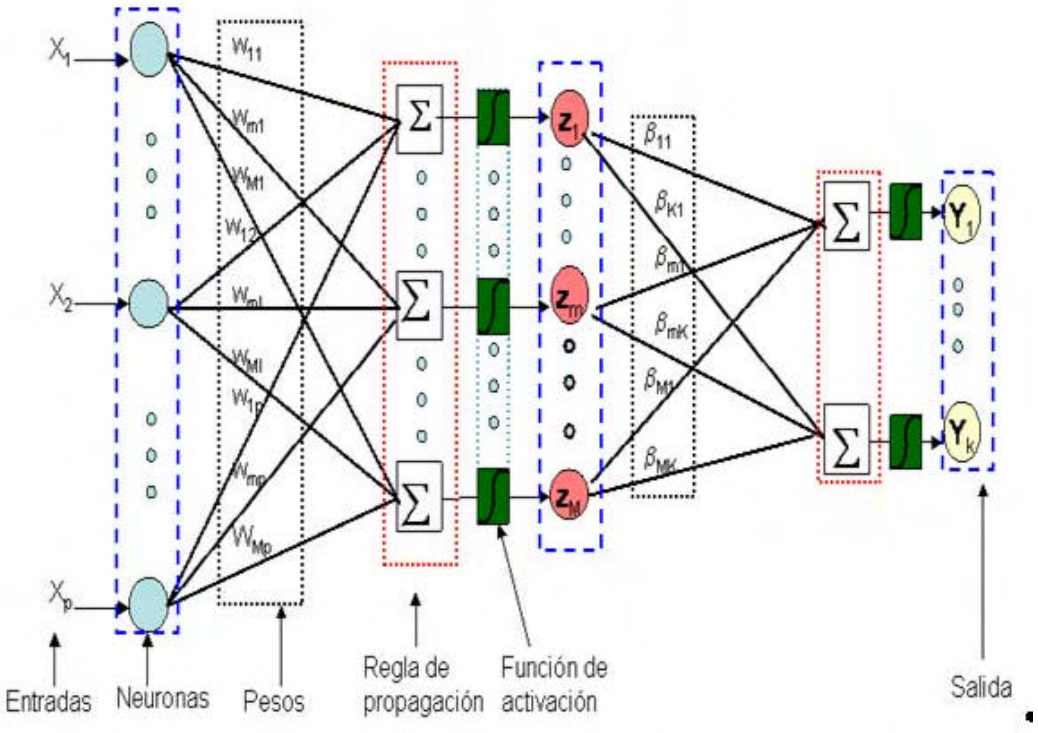


Figura 1.1. Representación de una red neuronal.

Antes de definir el siguiente elemento de la red neuronal, se define la notación para la realización del siguiente trabajo. X representa el vector aleatorio, sus componentes se denotan por X_j y la salida puede denotarse por Y si es cuantitativa y G si es categórica. Una observación de X puede escribirse por x_i.

Entradas.- Información entrante, X_1, $X_{2,...,}$ X_p o vector aleatorio $X^T = (X_1, X_2, ..., X_p)\, \varepsilon \Re^P$

Pesos (parámetros). En cada neurona, a las entradas se les asigna un peso, factor de importancia. Este peso es un número que se modifica durante el proceso de aprendizaje de la red neuronal. En los pesos se almacena la información que hará que la red sirva para un propósito u otro. Los pesos para la capa oculta se denotara con w y para la capa de salida con b.

Regla de propagación. Con las entradas y los pesos se suele hacer diferentes tipos de operaciones. Una de las operaciones más comunes es sumar las entradas, pero teniendo en cuenta la importancia de cada una de estas. A esta combinación de entradas con los pesos se le denomina Net. La Net para la capa oculta y para la capa de salida está definida respectivamente por:

$$Net_m = w_m^T X + w_0 \quad \text{y} \quad Net_k = b_k^T Z + b_0 \tag{1.1}$$

Donde $w_m^T = \left(w_{1m}, w_{2m}, ..., w_{pm}\right)$ y $b_k^T = \left(b_{1k}, b_{2k}, ..., b_{Mk}\right)$

Función de activación. El valor obtenido a partir de la regla de propagación (Net), se filtra a través de una función llamada ''función de activación'' o ''función de transferencia''. Esta función proporciona la salida de la red.

La función de activación para la capa de salida se denotara por $g(\bullet)$ y para capa oculta con $s(\bullet)$. La forma de estas funciones la elige el investigador de acuerdo al objetivo que se traza.

Las tres principales funciones de transferencia son las siguientes:

a) Función Escalón.

La función escalón se asocia a redes cuya salida es binaria, en las cuales cuando la Net es mayor o igual que cero, la activación es 1. Si la Net es menor que 0, la activación es 0.

$$g(Net) = \begin{cases} 1 & si \quad Net \geq 0 \\ 0 & si \quad Net < 0 \end{cases} \tag{1.2}$$

b) *Función Identidad*

La función identidad equivale a no aplicar la función activación y está definida como:

$$g(Net) = Net \tag{1.3}$$

Su uso es muy limitado.

c) *Función Exponencial.*

La función exponencial es definida como.

$$g(Net) = \frac{1}{1+\exp(-Net)} \tag{1.4}$$

La importancia de la función exponencial es que su derivada está definida en todo $\Re$ y es siempre positiva y cercana a cero para los valores grandes positivos o grandes negativos, esto es particularmente útil para definir métodos de aprendizaje en los cuales se usa derivadas.

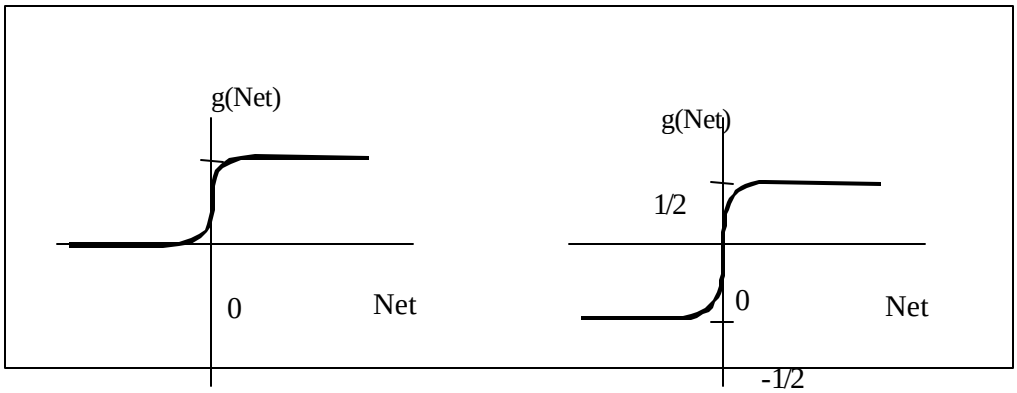


Figura *1.2. Funciones de Activación exponencial.*

Salida. Es el resultado de la función de transferencia

$$f(X) = g(Net) \tag{1.5}$$

1.4 Arquitectura de la Red.

Las neuronas se organizan en una secuencia de capas con un determinado patrón de interconexiones entre los diferentes elementos dependiendo de la posición de las capas, éstas pueden ser de entrada, oculta o de salida. La manera como se interconectan las neuronas puede ser:

- Conexiones hacia delante (feedforward), de la capa de entrada a la de salida. (se usa en aplicaciones de clasificación, regresión, aproximación de funciones).
- Conexiones laterales dentro de la misma capa. (se usan para formar conglomerados y para la extracción de componentes principales)
- Conexiones con retardo temporal. (con aplicaciones en predicción y clasificación).

Antes de definir el modelo estadístico de una red neuronal, introduciremos algunos conceptos de modelos aditivos generales.

1.5 Modelos Aditivos Generales.

Una herramienta muy importante en el análisis de datos es el modelo de regresión lineal. En el caso más simple el objetivo es describir la dependencia de la media de Y (variable dependiente) como función de X (variable independiente). Bajo el supuesto de que la media de Y es función lineal de X.

$$E(Y/X) = \boldsymbol{a} + X\boldsymbol{b} \tag{1.6}$$

Estos modelos lineales presentan limitaciones en describir problemas de la vida real donde los efectos son no lineales. Los modelos estadísticos más generales y flexibles que pueden ser usados para identificar y caracterizar efectos no lineales, son los **Modelos Aditivos Generales (GAM)** [11] que tienen la forma:

$$h(E(Y \mid X_1,..., X_p)) = a + \sum_{l=1}^{p} f_l(X_l) = a + f_1(X_1) + ... + f_p(X_p) \quad (1.7)$$

Donde

$X_1,..., X_p$ son variables independientes e Y es la variable dependiente

Las f_l son funciones no parametricas (smooth) no especificadas. La forma arbitraria de estas funciones hace que el modelo sea más flexible.

Un caso particular del modelo aditivo general es el **modelo de regresión logístico aditivo para dos clases**. Este modelo está definido por:

$$\log\left(\frac{m(Y)}{1-m(Y)}\right) = a + \sum_{l=1}^{p} f_l(X_l) \quad (1.8)$$

Donde $m(Y) = P(Y = 1/ X)$

En general, la condición media $m(Y)$ de la respuesta está relacionado con una función aditiva de los predictores mediante la función enlace h.

$$h\left[m(Y)\right] = a + \sum_{l=1}^{p} f_l(X_l) \quad (1.9)$$

Para el ajuste de modelos aditivos:

$$Y = a + \sum_{l=1}^{p} f_l(X_l) + e \text{ , con } E(e) = 0 \text{ y } Var(e) = s^2, \quad (1.10)$$

existen varios métodos. Uno de ellos es el **algoritmo ''backfitting''** [11] que consiste en estimar las funciones $f_1,...., f_p$, en forma iterativa y con el siguiente algoritmo:

……

Algoritmo 1.1: Para estimar un modelo aditivo general de la forma $Y = a + \sum_{l=1}^{p} f_l(X_l) + e$, se debe primero :

Inicializar : $\hat{a} = \frac{1}{N}\sum_{i=1}^{N} y_i,\ \hat{f}_l \equiv 0 \quad \forall i, l.$

1. repetir varias veces desde l = 1, 2, …, p

$$\hat{f}_l <--- S_l\left[\{y_i - \hat{a} - \sum_{k \neq l} \hat{f}_k(x_{ik})\}_1^N\right]$$

$$\hat{f}_l <--- \left[\hat{f}_l - \frac{1}{N}\sum_{i=1}^{N} \hat{f}_l(x_{il})\right]$$

$S_l = X_l^{'}\left[X_l^{'} X_l + l\, I\right]^{-1} X_l$, es la matriz smother.

l : parámetro smothing.

Este algoritmo termina cuándo la función $\hat{f}_l$ se estabiliza (no varia).

……

Si el modelo aditivo general $Y = a + \sum_{l=1}^{p} f_l(X_l) + e$, $e \approx N(0, s^2)$ es correcto para algún k entonces las funciones $f_k(x)$ pueden estimarse mediante $f_k(X_k) = E\left(y - a - \sum_{l \neq k} f_l(X_l) / X_k\right)$, esta estimación se realiza en forma secuencial para todas las funciones $f_l(X_l)$.

Existen varias clases de modelos aditivos generales. En el contexto de las redes neuronales existen dos clases especiales de modelos aditivos generales que son muy importantes:

La primera clase especial de modelos aditivos, se tiene cuando las funciones arbitrarias f_l, toman la forma:

$$f_l(X_l) = \sum_{m=1}^{M_l} b_{lm} h_{lm}(X_l). \qquad (1.11)$$

Donde h_{lm} es una transformación de X_l.

A estas funciones arbitrarias se les denomina **"expansiones de funciones base lineal en X_l"** [9] . En este caso el modelo aditivo toma la forma:

$$f(X) = \sum_{l=1}^{p} f_l(X_l) = \sum_{l=1}^{p} \sum_{m=1}^{M} b_{lm} h_{lm}(X_l) \qquad (1.12)$$

Si nosotros modelamos cada una de las f_l usando expansión de funciones base, entonces el modelo puede ser ajustado por mínimos cuadrados

Una segunda clase particular de modelo aditivo general es llamado **"projection pursuit regression(PPR)"** [9] **.** Este modelo se obtiene cuando:

$$f(X) = \sum_{m=1}^{M} g_m(a_m^T X) \qquad (1.13)$$

$a_m^T X$, son combinaciones lineales de las entradas que representan la proyección de X sobre el vector a_m .

Para ajustar un modelo PPR, dado un conjunto de entrenamiento E = {(x_i, y_i) / x_i =(x_{i1},…$x_{il,...}$, x_{ip}) ,i=1,…N }, se minimiza " la función de error"

$$R = \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} g_m(a_m^T x_i) \right)^2 \qquad (1.14)$$

Donde $x_i \in \Re^p$ es una observación de $X = (X_1,..., X_p)$

1.6 **Redes Neuronales.**

Dado el vector aleatorio $X \in \Re^p$ y la variable dependiente $Y \in \Re$, una red neuronal es un modelo estadístico no lineal [7], definido con la relación:

$$Y = f(X) + e = g^s\left[b_{0k} + b_k^T s\left(w_{0m} + w_m^T X\right)\right] + e \qquad (1.15)$$

Donde

$g^s = g \circ g \circ g \circ ... g$, se construye de forma recursiva (composición de la función g, s-veces) a partir de aproximaciones no-lineales.

g^s , s son funciones de activación o transferencia.

b, w son vectores de pesos.

s : es el número de capas ocultas.

Frecuentemente en muchos problemas, la red de una sola capa oculta es suficiente para describir el comportamiento de los datos. El modelo estadístico no lineal para este caso, está definido por:

$$y = f(X) + e = g\left[b_{0k} + b_k^T s\left(w_{0m} + w_m^T X\right)\right] + e \text{ , cuando } s = 1. \qquad (1.16)$$

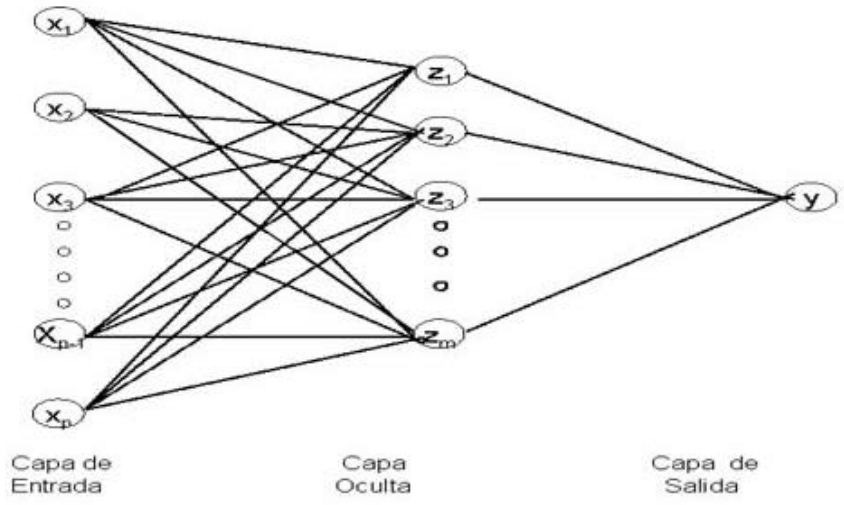


Figura 1.3. Representación de la Red Neuronal con una capa oculta.

Cada salida Z_m, para cada unidad de la capa oculta de la red, en la figura 1.3,

se puede escribir como:

$$Z_m = s\left(w_{0m} + w_m^T X\right) = s\left(Net_m\right) \text{ m=1,..M} \qquad (1.17)$$

Una salida simple de la red está dada por:

$$y = f(X) = g(b_{0k} + b_k^T s\left(w_{0m} + w_m^T X\right)) = g(b_{0k} + b_k^T Z_m) = g\left(Net_k\right), \text{k=1} \qquad (1.18)$$

Donde s y g son funciones de transferencia para la capa oculta y para la capa de salida, respectivamente Net_m y Net_k son funciones lineales en X y en Z respectivamente.

La expresión de f como una función de X es una función de regresión no lineal muy complicada, con vector de parámetros w, b .

1. La red neuronal de una capa oculta es exactamente un modelo aditivo PPR, cuando: g es una función identidad, $b_{0k} = w_{0m} = 0$; bajo estas condiciones la relación (1.18) está dada por:

   $$y = f(X) = b_k^T Z_m = \sum_{k=1}^{1}\sum_{m=1}^{M} b_{km} Z_m = \sum_{m=1}^{M} b_m Z_m = \sum_{m=1}^{M} b_m s\left(w_m^T X\right)$$

   En este caso particular la red posee el modelo siguiente:

   $$f(X) = \sum_{m=1}^{M} b_m Z_m = \sum_{m=1}^{M} b_m s(w_m^T X) = \sum_{m=1}^{M} g_m(a_m^T X) \qquad (1.19)$$

   Donde $g_m(a_m^T X) = b_m s(w_m^T X)$.

2. Si g es una función identidad, $M = p$, $b_{0k} = w_{0m} = 0$ y b_k, w_m vectores unitarios, bajo estas condiciones el modelo de la red de una capa oculta es exactamente un modelo aditivo general dado por:

   $$y = f(X) = \sum_{k=1}^{1}\sum_{m=1}^{M} b_{km} Z_m = \sum_{m=1}^{M} b_m s\left(w_m^T X\right) = \sum_{m=1}^{p} s_m\left(X_m\right) = \sum_{m=1}^{p} f_m(X_m) \qquad (1.20)$$

   En este caso particular la red posee el modelo siguiente:

$$f(X) = \sum_{m=1}^{p} s_m(X_m) = \sum_{m=1}^{p} f_m(X_m), \qquad (1.21)$$

donde $f_m(X_m) = s_m(X_m)$

En resumen podemos definir la red neuronal como un modelo aditivo general, caso 2. En particular una red neuronal se puede ver como un PPR caso1. Además puede ser vista como una adaptación de funciones base cuando M=1 , g función de identidad y $b_{0k} = w_{0m} = 0$ en este caso el modelo tiene la forma:

$$f(X) = \sum_{l=1}^{p} \sum_{m=1}^{M} b_{lm} h_{lm}(X_l) = \sum_{l=1}^{p} b_l s(w_l^T X + w_0) \qquad (1.22)$$

1.7 **Aprendizaje**

La principal característica importante de las redes neuronales es su capacidad para ‘‘aprender de su ambiente’’ a través de un proceso interactivo de ajustes de sus pesos. Este proceso de adaptación o modificación de los pesos de acuerdo con el entorno en el que se encuentra sumergida la red recibe el nombre de ***regla de aprendizaje*** o simplemente aprendizaje y su trascripción en forma de procedimiento se denomina **algoritmo de aprendizaje** [4].

El aprendizaje puede ser supervisado o no supervisado. En el presente trabajo se usará el aprendizaje supervisado.

En el aprendizaje supervisado la base de datos se divide en tres conjuntos: de entrenamiento, validación y prueba. Los datos de entrenamiento se usa para ajustar el modelo, los datos de validación para estimar el ‘‘error del modelo’’ y los datos de prueba para estimar la bondad del modelo. Típicamente se considera el 50% de los datos para entrenamiento, 25% para validación y 25% para prueba.

Para permitir a la red neuronal el aprendizaje de un comportamiento deseado, se alimenta a la red con los datos de entrenamiento.

$E=\{(x_i, y_i)/ \quad x_i=(x_{i1},\ldots x_{il\ldots}, x_{ip}) \quad i=1,\ldots N\}$ (1.23)

Donde en términos de redes $x_i \varepsilon \Re^P$ representan las entradas e $y_i \varepsilon \Re$ las salidas.

Comparando la salida de la red $f(X)=g(Net)$ con la salida de los datos de entrenamiento (y_i) ; se obtiene una diferencia que se usa para calcular una función de pérdida la que penaliza el error de la respuesta de la red. Con un algoritmo apropiado es posible modificar los valores de los pesos con el fin de reducir el error. Estas correcciones deben realizarse varias veces, para todo el conjunto de observaciones de entrenamiento.

En general, el problema de aprendizaje, desde el punto de vista estadístico, se puede ver como estimación de los parámetros (pesos) y ajuste del modelo. Los vectores de pesos w y b son elegidos de tal manera que alguna medida de ajuste sea mínima. Dado el modelo de la red.

$$y = f(X) + e \,. \qquad (1.24)$$

El objetivo del aprendizaje en redes neuronales es obtener una aproximación de $f(X)$ para todo los valores de X, dado el conjunto de entrenamiento E. Una clase de aproximación para redes neuronales puede ser expresada como:

$$f(X) = \sum_{m=1}^{M} b_m s_m(X) \qquad (1.25)$$

donde s_m son funciones o transformaciones del vector de entrada X, tradicionalmente son polinomios o expansiones trigonométrica. También podemos encontrar expansiones no lineales, tales como $s_m = \frac{1}{1+\exp(-X^T w_m)}$.

Este tipo de expansión se llama ‘’expansion de funciones de base”.

Para estimar los parámetros en f , se define una función de pérdida $L(y_i, f(x_i))$, que penaliza el error de predicción. Esta función es minimizada para el conjunto de datos de entrenamiento:

$$R(\bullet) = \sum_{i=1}^{N} L(y_i, f\ (x_i)) \qquad (1.26)$$

Utilizando algún criterio para la función de pérdida.

1.7.1 **Aprendizaje supervisado**

Este tipo de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada. El supervisor comprueba la salida de la red y en el caso de que ésta no coincida con la salida correcta se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la correcta.

1.7.2 **Aprendizaje no-supervisado**

Es cuando no existe un agente externo que indique si la respuesta es la adecuada para una observación de entrenamiento

1.8 Tipos de problemas que se resuelven con las Redes Neuronales.

Las redes neuronales se pueden usar como métodos estadísticos para problemas de clasificación, regresión, análisis por conglomerados (aprendizaje no supervisado), componentes principales, predicción y otros.

La presente tesis está enfocada a la combinación de predicciones de clasificadores, por lo tanto, nosotros abordaremos solo los problemas de regresión y clasificación por ser de interés para este trabajo. Inicialmente

presentamos una introducción de la regresión y clasificación y en (1.11.3) se desarrolla el proceso de estimación mediante redes neuronales.

1.1.1 **Regresión**.-

Dado el vector de entrada $Xe\Re^p$ y el vector de salida $Ye\Re$, el modelo de regresión está definido por:

$$Y = f(X) + e \tag{1.27}$$

El objetivo es seleccionar f para predecir Y ,dado algún valor de X.

Para el ajuste del modelo se compara la salida real con el valor obtenido por la red $f(X)$, mediante una medida, llamada función de pérdida $L(Y, f(X))$ que penaliza el error de predicción. La función de pérdida para la regresión es comúnmente el error cuadrático:

$$L(Y, f(X)) = (Y - f(X))^2 \tag{1.28}$$

Para estimar f , previamente definimos el error esperado de predicción, como la esperanza de $L(Y, f(X))$, esto es:

$$EPE(f) = E[L(Y, f(X))] = E[Y - f(X)]^2 \tag{1.29}$$

$$= \int (y - f(x))^2 P(dx, dy)$$

$$= \int (y - f(x))^2 P(dy/dx) P(dx)$$

$$EPE(f) = E_X E_{y/X} \left([Y - f(X)]^2 / X \right)$$

$$f(x) = \arg\min_c E_{y/X} \left([Y - c]^2 / X = x \right) \tag{1.30}$$

La solución es:

$$f(x) = E(y / X = x) \tag{1.31}$$

1.1.2 **Clasificación.**

Se tiene un problema de clasificación, cuando la salida es una variable categórica GεC, que denotaremos con G en lugar de Y, G asume valores discretos en un conjunto finito C = {1,2,.., K} donde los números son etiquetas que representan clases $\{G_1, G_2, .., G_K\}$.

Una regla de clasificación o clasificador se denotará por G(X), donde G(X)=k, significa que un objeto con vector de características es asignado a la k-ésima clase G_k. Más específicamente con esta regla lo que intenta es construir una partición del espacio de características en K regiones $R_1, .., R_K$ mutuamente excluyentes.

En el caso de clasificación [9], se requiere de una función de pérdida diferente al definido en el caso de regresión. La función de pérdida más usada para la clasificación es 0-1, definida por:

$$L(G, G(X)) = I(G \neq G(X)) = \begin{cases} 1 & G \neq G(X) \\ 0 & G = G(X) \end{cases} \tag{1.32}$$

Y el valor esperado del error de predicción mediante:

$$EPE = E[L(G, G(X)] \tag{1.33}$$

Donde la esperanza es tomada con respecto a la distribución conjunta $P(G, X)$.

$$EPE = \sum_{k=1}^{K} L(G_k, G(X)) P(G_k / X) P(X) = E_X \sum_{k=1}^{K} L(G_k, G(X)) P(G_k / X), \tag{1.34}$$

minimizando el valor esperado de predicción se estima el clasificador

$$\hat{G}(X) = \arg\min_{g \in C} \sum_{k=1}^{k} L(G = k, g) P(G = k / X = x), \tag{1.35}$$

simplificando como la función pérdida asume valores 0 o 1 se tiene:

$$\hat{G}(X) = \arg\min_{g \in C} [1 - P(G = g / X = x)] \tag{1.36}$$

$$\hat{G}(X) = \arg\max_{g \in C} \left[P(G = g / X = x) \right] \quad (1.37)$$

$$\hat{G}(X) = k \quad si \quad P(G = k / X = x) = \max_{g \in C} P(G = g / X = x) \quad (1.38)$$

A la solución dada en (1.38), se le llama clasificador de Bayes.

En resumen, el problema de clasificación consta de los siguientes pasos.

a) Dado un conjunto de entrenamiento E de N observaciones, cada observación X_i en E pertenece a una de las K clases $\{G_1, \ldots, G_K\}$.

b) Con el conjunto de E, se construye una función G(X): $\Re^p \rightarrow \{G_1, \ldots, G_k\}$ a esta función se le denomina clasificador.

c) El clasificador G(X) asigna a la observación X , a una de las clases G(X) å{G $_1$,…, G$_k$}

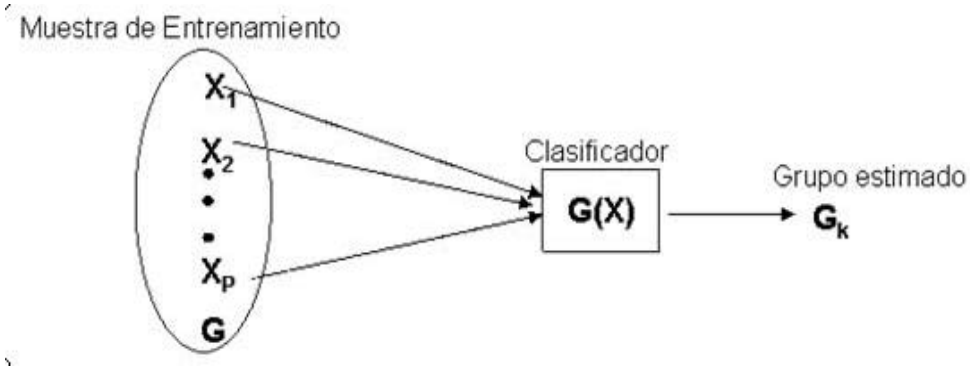


Figura 1.4. Representación de un Clasificador.

1.2 **Perceptrón.**

Este fue el primer modelo de red neuronal desarrollado por Rosenblatt en 1958-1962. Despertó enorme interés en los años de 60, debido a su capacidad para aprender a reconocer comportamientos sencillos [4]. El ''Perceptron'' es una red neuronal simple, auque no por ello resulta exento de utilidad ya que, a pesar de sus inherente limitación es un procesador, eficiente de información.

Por sus características tales como comportamiento geométrico y estadístico, algoritmos de aprendizaje, el balance que se debe efectuar en su diseño entre el poder de aproximación y la estimación son extensibles a otros sistemas neuronales más complejos, su estudio como entidad independiente esta plenamente justificada.

1.2.1 **Definición.-**

Dado $X \in \Re^p$ e $Y \in \Re$, el perceptrón es la red neuronal con una neurona en la capa oculta (fig. 1.5) y que está definida mediante la relación:

$$y = f(X) + e = g(w_0 + w^T X) + e \quad , Net = w_0 + w^T x \tag{1.39}$$

Este es el modelo más simple de red neuronal que consiste de una combinación lineal, formada por varias neuronas para recibir entradas a la red y una neurona de salida. La aplicación del perceptrón se limita a problemas linealmente separables. Es decir actúa como un clasificador lineal dado por:

$$g(Net) = \begin{cases} 0 & cuando \quad Net < 0 \\ 1 & cuando \quad Net \geq 0 \end{cases} \tag{1.40}$$

Dado un conjunto de entrenamiento x $\varepsilon \Re^P$, el propósito del perceptrón es de clasificar en uno de los dos Grupos G_1 o G_2. La regla de decisión para la clasificación es asignar x al grupo G_1 si la salida del perceptrón es 1 y al grupo G_2 si es 0.

1.2.2 **Arquitectura**

El perceptrón tiene una ‘’arquitectura’’ que se puede dividir en tres partes.

a) Eventualmente una serie de transformaciones o aplicaciones, con el espacio de entrada X (Pre-Procesado).
b) Una regla de propagación (Net) del vector de entrada.
c) Una transformación g realizada sobre Net.

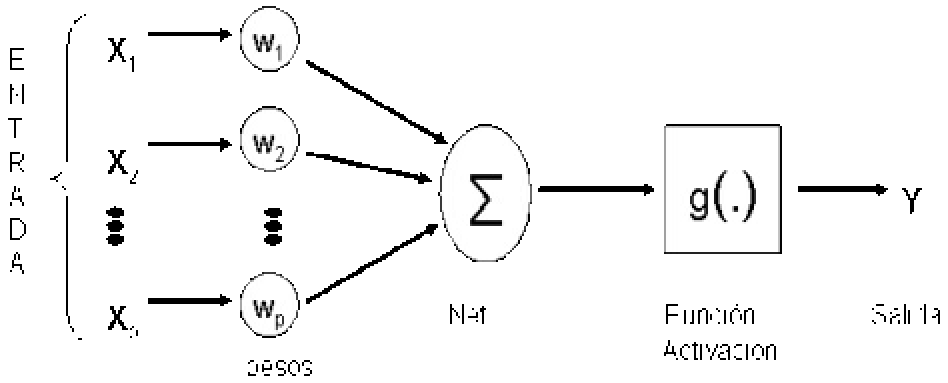


Figura 1.5. Representación del Perceptrón.

1.2.3 **Aprendizaje del Perceptron.**

El algoritmo de aprendizaje del perceptrón [4] trata de encontrar una separación del espacio de entrada mediante un hiperplano por minimización del error cuadrático medio definido por:

$$R = \sum_{i=1}^{N} L(y_i, f(x_i)) = \frac{1}{2}\sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} R_i \qquad (1.41)$$

donde: $f(x) = g(Net)$

R se usa como función de error para el conjunto de entrenamiento principalmente por la simplicidad analítica; sin embargo, puede tener otras expresiones, dependiendo de la aplicación que se tenga.

La función de error R es una función matemática definida en $\Re^p$ que para un conjunto de entrenamiento define una superficie que tendrá muchos mínimos locales y un mínimo global.

La actualización de los pesos se realiza con desplazamientos pequeños en el espacio de los pesos en la dirección en la cual *la función error decrece* más rápidamente, a este proceso de modificación de los pesos y obtención de la

información local de la superficie, se le denomina método de gradiente descendiente.

El gradiente se aplica a la función de error y en un punto particular mide la dirección del crecimiento máximo.

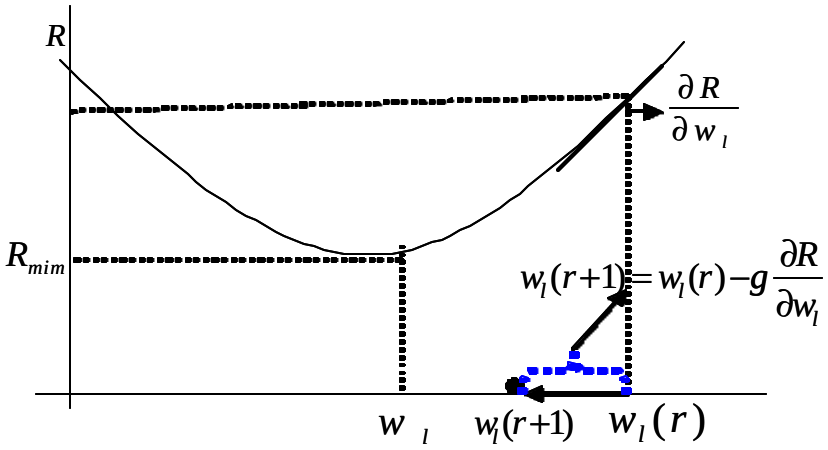


Figura 1.6 : Ilustración del método de gradiente descendiente.

La modificación de los pesos es proporcional al gradiente descendiente de la función error.

$$\Delta w_l = -g\frac{\partial R}{\partial w_l} = -g\sum_{i=1}^{N}\frac{\partial R_i}{\partial w_l} \quad l = 1,..,p \ , \ \Delta w_l = w_l(r+1) - w_l(r) \tag{1.42}$$

aplicando la regla de la cadena para el calculo de $\frac{\partial R_i}{\partial w_l}$

$$\frac{\partial R_i}{\partial w_l} = \frac{\partial R_i}{\partial Net}\frac{\partial Net}{\partial w_l} \tag{1.43}$$

y utilizando la función de transferencia identidad $g(Net) = Net$, se tiene:

$$\frac{\partial R_i}{\partial Net} = \frac{\partial}{\partial Net}\left[\frac{1}{2}(y_i - f(x_i))^2\right] = -(y_i - f(x_i)) = -R_i \tag{1.44}$$

y $$\frac{\partial Net}{\partial w_l} = \frac{\partial \sum_{l=1}^{p} w_l x_{il}}{\partial w_l} = x_{il} \qquad (1.45)$$

Reemplazando (1.45) y (1.44) en (1.43) se tiene.

$$\frac{\partial R_i}{\partial w_l} = -R_i x_{il} \qquad (1.46)$$

la actualización de los pesos en la r-ésima iteración para el conjunto de entrenamiento es:

$$w_l(r+1) = w_l(r) + g \sum_{i=1}^{N} R_i x_{il} \qquad (1.47)$$

g : se llama factor de aprendizaje . Este factor varía entre 0 y 1.

La finalidad del aprendizaje del perceptrón, es encontrar una separación lineal del espacio de entrada.

En el apéndice se muestra algunos métodos clásicos de discriminación lineal como perceptrón.

1.3 **Red Neuronal Multicapa**

El perceptrón, sólo puede solucionar problemas linealmente separables que, obviamente, resultan ser casos particulares de los problemas reales. La pregunta que surge es si es posible la combinación en varias capas de elementos de proceso tipo perceptrón, y si es así, cómo se podría entrenar la red. Lo que se buscaría con esta combinación de ‘’perceptrónes’’ en varias capas es la división del espacio de entradas en varios subespacios, cada uno de ellos representando una característica del problema a tratar. Luego, en capas posteriores se irían delimitando más detalladamente las distintas zonas del espacio de características, así hasta conseguir la salida (partición del

espacio de entradas) deseada. De esta manera situando los perceptrones en cascada mediante una estructura de múltiples capas podemos implementar fronteras de decisión mucho más complejas [4], como se ilustra en la figura 1.7.

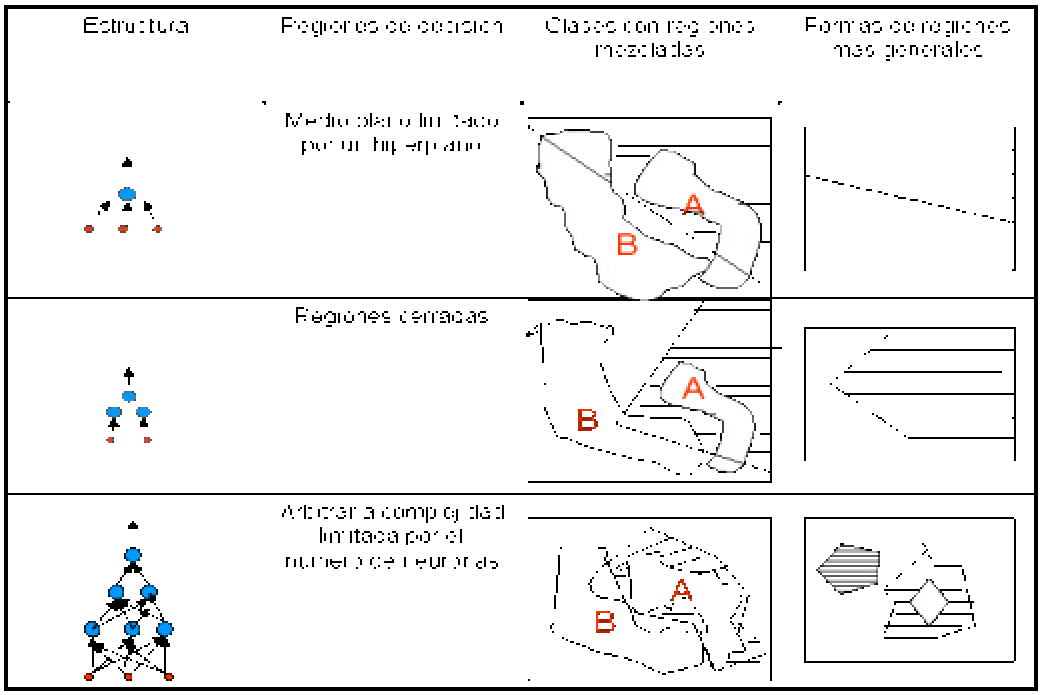


Figura 1.7: Fronteras de decisión.

1.1.1 **Antecedentes**

Rosenblatt (1962) y **Minsky- Papera,** analizaron la posibilidad de aumentar la funcionalidad del perceptrón usando una o más capas ocultas [4]. El primer algoritmo de entrenamiento para redes multicapa fue desarrollado por Paul Werbos en 1974, esté se desarrolló en un contexto general, para cualquier tipo de redes.

La regla de aprendizaje para una red de tipo Feedforward (Hacia delante) compuesta de una o mas capas ocultas de forma que las neuronas de una misma capa no esten conectadas entre si, fué propuesta por **Rumelhart** y de forma independiente por **Parker** y **Le Cun**. Esta regla se conoce como regla de *propagación hacia atrás* (**BACKPROPAGATION**).

1.1.2 **Estructura de la Red Multicapa.**

La estructura típica de una red multicapa se observa en la figura

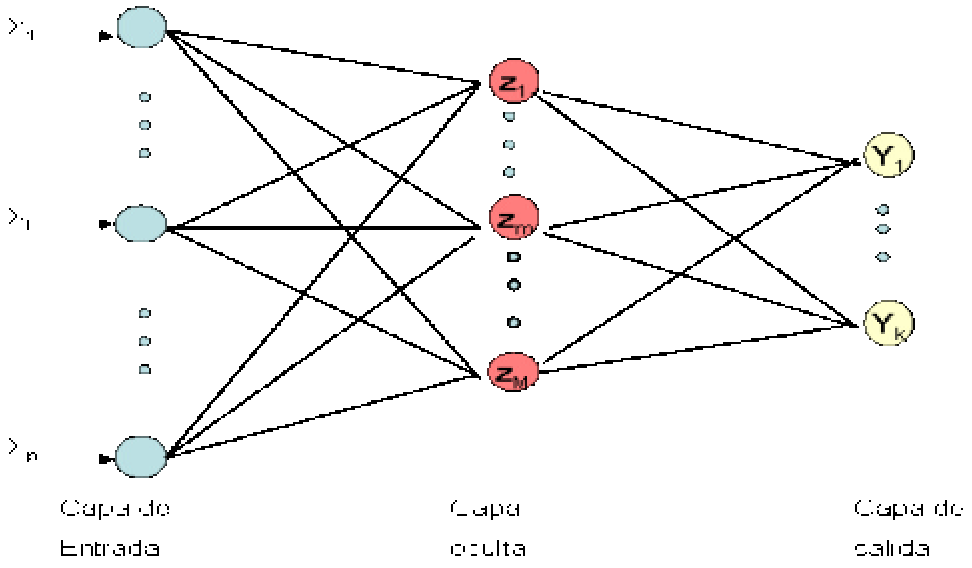


Figura 1.8. Representación de la Red Multicapa.

La primera capa es llamada capa de entrada que toma datos del conjunto de entrenamiento E y la última capa es denominada de salida, cuyos pesos adquieren, mediante un proceso de aprendizaje, distintos valores para observaciones distintas.

1.1.3 **Backpropagation.-**

Este es un algoritmo de aprendizaje supervisado, que emplea un ciclo de propagación (adaptación de dos fases). Aplicada una observación a la entrada de la red, como estímulo, ésta se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. Esta salida se compara con la salida real y se calcula la señal de error para cada una de las observaciones. Los errores se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo; las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todas las observaciones de entrenamiento.

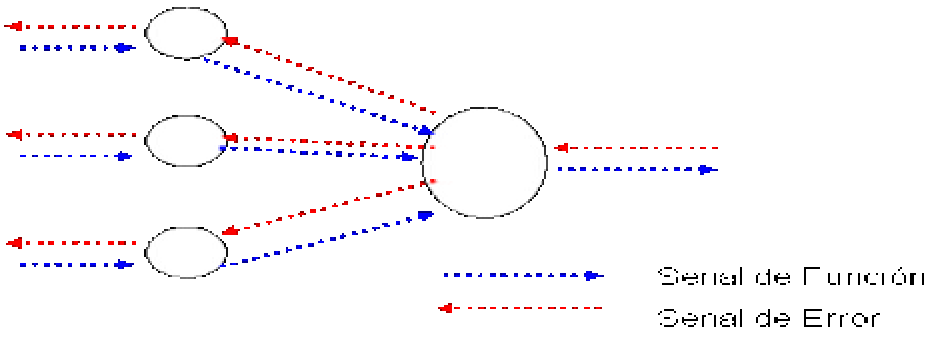


Figura 1.9. Propagación de las señales en un Perceptrón Multicapa.

La importancia de este proceso consiste en que a medida que se entrena la red las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada.

Varias investigaciones han demostrado que durante el proceso de entrenamiento la red Backpropagation tiende a desarrollar relaciones internas entre neuronas con el fin de organizar los datos de entrenamiento en clases [13].

1.1.3.1 **Regla de aprendizaje de Redes Multicapa.**

El entrenamiento de una red neuronal multicapa se realiza mediante un proceso de aprendizaje. Para realizar este proceso se debe tener inicialmente definida la topología de la red, esto es el número de neuronas en la capa de entrada, el cual depende del número de componentes del vector de entrada, de la cantidad de capas ocultas y el número de neuronas de cada una de ellas, del número de neuronas en la capa de la salida, que depende del número de componentes del vector de salida o patrones objetivo y las funciones de transferencia requeridas en cada capa. Con base en la topología escogida se asignan valores iniciales a cada uno de los pesos (parámetros) que conforma la red.

Es importante recalcar que no existe una técnica para determinar el número de capas ocultas, ni el número de neuronas que debe contener cada una de ellas para un problema específico, esta elección es determinada por la experiencia del experimentador [9].

Cada observación de entrenamiento se propaga a través de la red para producir una respuesta en la capa de salida, la cual se compara con la salida real para calcular el error en el aprendizaje, este error marca el camino más adecuada para la actualización de los pesos que al final del entrenamiento producirán una respuesta satisfactoria a todas las observaciones de entrenamiento. Esto se logra minimizando el error en cada iteración del proceso de aprendizaje.

Desarrollamos este procedimiento para una red con una capa de entrada, una oculta, una de salida. La salida de la red está dada por:

$$f_k(X) = g\left[b_{0k} + b_k^T s\left(w_{0m} + w_m^T X\right)\right] = g(Net_k) \qquad (1.48)$$

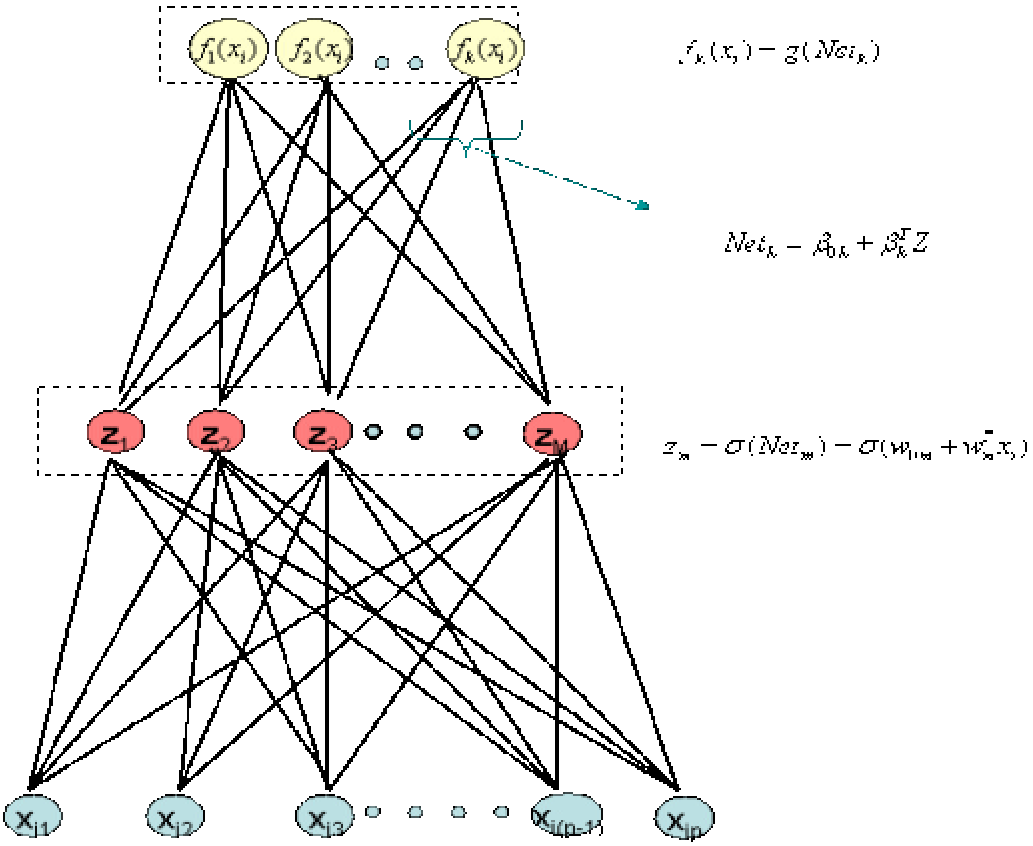


Figura 1.10. Representación de la red de una capa de entrada, una oculta y una salida.

Para iniciar el entrenamiento se le presenta a la red una observación x_i del conjunto de entrenamiento E = {(x_i, y_i) / i=1,…N } la cual tiene p componentes.

$$x_i^T = (x_{i1}, \ldots, x_{il}, \ldots, x_{ip}) \qquad (1.49)$$

Cuando se le presenta a la red una observación de entrenamiento, está se propaga a través de las conexiones existentes produciendo una Net en cada una de las M neuronas de la capa oculta mediante:

$$Net_{mi} = w_m^T x_i + w_{0m} = \sum_{l=1}^{p} w_{ml} x_{il} + a_{0m}, \quad m = 1, \cdots M, \quad i = 1, ..., N, \quad w_m^T = (w_{m1}, .., w_{ml}, ., w_{mp}), \quad (1.50)$$

Donde

w_{ml}: peso que corresponde al componente l de la entrada con la neurona m de la capa oculta.

w_{0m}: umbral (Ganancia) de la neurona m de la capa oculta.

Cada una de las neuronas de la capa oculta tiene como salida Z_{mi}.

$$Z_{mi} = s\,(Net_{mi}) = s\,(\sum_{l=1}^{P} w_{ml} x_{il} + w_{om}), \qquad (1.51)$$

s : función de transferencia de las neuronas de la capa oculta.

Las salidas Z_{mi} de las neuronas de la capa oculta (de M componentes) son las entradas a los pesos de conexión de la capa de salida para la i-ésima

observación del conjunto de entrenamiento, este comportamiento esta descrito por la siguiente ecuación

$$Net_{ki} = b_{0k} + b_k^T Z_i = \sum_{m=1}^{M} b_{km} Z_{mi} + b_{0k}, \quad b_k = (b_{1m},..., b_{km},..., b_{kM}) \tag{1.52}$$

b_{km}: peso que corresponde a la neurona m de la capa oculta con la neurona k de la capa de salida, la cual cuenta con M neuronas.

Z_{mi}: salida de la neurona m de la capa oculta, la cual cuenta con M neuronas.

b_{0k}:ganancia de la neurona k de la capa de salida.

Net_{ki}: entrada neta a la neurona k de la capa de salida

La red produce una salida final para cada observación de entrenamiento descrita por la ecuación

$$f_k(x_i) = g(Net_{ki}) \qquad k=1,..,K \tag{1.53}$$

g : función de transferencia de las neuronas de la capa de salida y K es el número de neuronas en la capa de salida.

Reemplazando (1.52) en (1.53) se obtiene la salida de la red en función de la entrada neta (Net_{ki}) y de los pesos de conexión con la ultima capa oculta.

$$f_k(x_i) = g(Net_{ki}) = g(b_{0k} + b_k^T Z) = g\left(b_{0k} + b_k^T s\left(w_{0m} + w_m^T x\right)\right) \tag{1.54}$$

La salida de la red $f_k(x_i)$ se compara con el valor real y_{ik} para calcular una medida de error llamado función de pérdida $L(y_{ik}, f_k(x_i))$, para penalizar el

error de predicción. Cuando la salida y_{ik} es una variable aleatoria continua, la función de pérdida comúnmente está dada por el error cuadrático:

$$L(y_{ik}, f_k(x_i)) = \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2 \qquad (1.55)$$

Cuando la salida G es una variable categórica, se requiere de otra función de pérdida denominada cross-entropía o deviance definida por:

$$L(G_{ik}, f_k(x_i)) = -\sum_{k=1}^{K} I(G_{ik} = k) \log f_k(x_i) = -\sum_{k=1}^{K} G_{ik} \log f_k(x_i) \qquad (1.56)$$

Donde $f_k(x)$ representa probabilidades, $f_k(x) = p_k(x)$, y está dada por la salida de la red.

La entropía mide la cantidad de información que existe en el proceso de aprendizaje.

El ajuste de una red neural multicapa tiene como objetivo la minimización de la función de error;

$q = \arg\min_q R(q)$, donde:

$$R(q) = \sum_{i=1}^{N} L(y_{ik}, f_k(x_i)) \;\; y \;\; q = (w_{ml}, w_{0m} / m = 1,...M; b_{km}, b_{k0} / k = 1,...,K) \qquad (1.57)$$

Los valores de los pesos que minimizan la función de error son obtenidos por el método de gradiente descendiente, que a continuación se describe:

1.1.1.1 **Gradiente descendiente.**

El espacio de los pesos genera una superficie de error que tendrá mínimos locales y mínimo global. Al evaluar el gradiente del error en un punto de esta superficie se obtendrá la dirección en la cual la función del error tendrá un mayor crecimiento. Como el objetivo del proceso de aprendizaje es minimizar el error, debe tomarse la dirección negativa del gradiente para obtener el mayor decremento del error y de esta forma su minimización, condición requerida para realizar la actualización de los pesos [12], [13].

La actualización de los pesos para la capa oculta y para la capa de salida en la r-ésima iteración están dadas [9], respectivamente por las ecuaciones:

$$w_{ml}^{(r+1)} = w_{ml}^{r} - g\sum_{i=1}^{N}\frac{\partial R_i}{\partial w_{ml}} \tag{1.58}$$

$$b_{km}^{(r+1)} = b_{km}^{r} - g\sum_{i=1}^{N}\frac{\partial R_i}{\partial b_{km}}, \tag{1.59}$$

donde:

g : "factor de aprendizaje", que controla la velocidad de aprendizaje.

Si queremos minimizar la función de error, deberíamos dirigirnos en contra de la gradiente (señala la dirección de crecimiento), lo que estamos haciendo es resbalar sobre la superficie del error, tratando de llegar al mínimo global de la superficie. Sin embargo, haciendo esto corremos el peligro de quedarnos atrapados en un mínimo local de la superficie.

Observaciones:

- Cuando se usa la técnica del gradiente descendiente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos; pues, al tener una información local de la superficie, no se sabe lo lejos o lo cerca que se está del punto mínimo. Con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo, mientras con incrementos pequeños, aunque se tarde más en llegar, se evita que esto ocurra. El elegir un incremento adecuado influye en la velocidad de convergencia del algoritmo, esta velocidad se controla a través del factor de aprendizaje g, el que por lo general se escoge como un número pequeño, para asegurar que la red encuentre una solución adecuada. Un valor pequeño de g significa que la red tendrá que hacer un gran número de iteraciones, si se toma un valor muy grande, los cambios en los pesos serán muy grandes, avanzando muy rápidamente por la superficie de error, con el riesgo de saltar el valor mínimo del error y permanecer oscilando alrededor de él, y sin poder alcanzarlo.

  Es recomendable aumentar el valor de g a medida que disminuye el error de la red durante la fase de entrenamiento, para garantizar así una rápida convergencia, teniendo la precaución de no tomar valores demasiado grandes que hagan que la red oscile alejándose demasiado del valor mínimo.

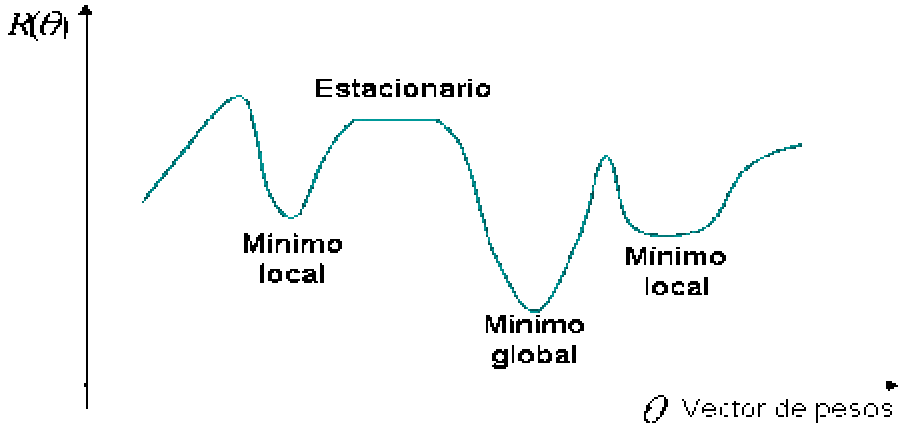


Figura 1.11. Forma ilustrativa de una superficie de error

- Cuando se encuentra un mínimo mediante el algoritmo de backpropagation, no se asegura que éste sea global. Sin embargo, cuando estimamos la red, usamos el aprendizaje bajo este error aun cuando esté sea alto.
- El algoritmo de backpropagation es muy sensible a la elección del factor de aprendizaje. Si éste es muy pequeño (cercano a cero) la variación de los pesos también será pequeña y la búsqueda pueda ser muy lenta, en cambio, si este factor es demasiado grande (cercano a 1), el valor de $R(q)$ puede descender describiendo grandes oscilaciones y saltando un mínimo global a un mínimo local. Esto puede evitarse añadiendo un término $m\Delta w_{(r)}$ llamado momentum en la ecuación $\Delta w = -g \frac{\partial R}{\partial w}$ [12].

$$\Delta w_{(r+1)} = -g \frac{\partial R}{\partial w} + m\Delta w_{(r)} \qquad (1.60)$$

La idea es dar a cada peso un impulso (inercia) de forma que la búsqueda del mínimo global continué.

Desarrollaremos el proceso de estimación para el caso de regresión y clasificación.

1.1.1.2 **Backpropagation para regresión.**

Consideremos una red neuronal como solución del problema de regresión.

Una red neuronal es un modelo no lineal para predecir el valor de la respuesta Y, basada en "p" medidas $X_1,\ldots,X_p$. El modelo está dado por:

$$Y = f(X) + e = g\left[b_{0k} + b_k^T s\left(w_{0m} + w_m^T X\right)\right] + e \text{, con } E(e) = 0 \text{ y } \text{var}(e) = s^2 \qquad (1.61)$$

En el caso de la regresión, la función g es la identidad y frecuentemente $s(Net) = 1/(1+\exp(-Net))$.

La salida esperada, está dada por:

$$E[Y/X] = f_k(X) = g\left[b_{0k} + b_k^T s\left(w_{0m} + w_m^T X\right)\right], \quad k=1,2\ldots,K \qquad (1.62)$$

Para el ajuste del modelo se compara la salida real (Y) con el valor estimado $f_k(X) = g\left[b_{0k} + b_k^T s\left(w_{0m} + w_m^T X\right)\right]$ por la red con K=1, mediante una medida determinada por la llamada función de pérdida $L(Y, f(X))$ que penaliza el error de predicción. La función de pérdida para la regresión comúnmente se define como:

$$L(Y, f_k(X)) = (Y - f_k(X))^2 \qquad (1.63)$$

Para estimar f, previamente definimos la función de error

$$R = \sum_{i=1}^{N} R_i = \sum_{i=1}^{N}\sum_{k=1}^{K} L(y_{ik}, f_k(x_i)) = \sum_{i=1}^{N}\sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2) \text{ ,} \qquad (1.64)$$

donde: $f_k(x_i)$ es el valor de $f_k(X)$ en la observación $x_i \in \Re^p$ de X.

La función de error (R) debe ser minimizada utilizando el proceso de aprendizaje de redes multicapa (Método de gradiente descendiente), la misma que se desarrollara a continuación para el caso de la regresión.

Actualización de los pesos en la capa de salida.

Para la actualización de los pesos en la capa de salida, utilizamos el método de gradiente descendiente.

$$\Delta b_{km} = -g \frac{\partial R}{\partial b_{km}} = -g \sum_{i=1}^{N} \frac{\partial R_i}{\partial b_{km}}, \qquad (1.65)$$

desarrollando:

$$\frac{\partial R_i}{\partial b_{km}} = \frac{\partial}{\partial b_{km}} \left[\sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2 \right]$$

$$\frac{\partial R_i}{\partial b_{km}} = -2(y_{ik} - f_k(x_i)) \frac{\partial (g(Net_{ki}))}{\partial (b_{km})} \ . \qquad (1.66)$$

Para calcular: $\frac{\partial (g(Net_{ki}))}{\partial (b_{km})}$, se debe utilizar la regla de la cadena, pues el error no es una función explícita de los pesos de la red, además la salida de la red $f_k(x_i) = g(Net_{ki})$ está explícitamente en función de Net_{ki} y de la ecuación (1.52) puede verse que Net_{ki} está explícitamente en función de b_{km} considerando esto se tiene la ecuación siguiente:

$$\frac{\partial g(Net_{ki})}{\partial b_{km}} = \frac{\partial (g(Net_{ki}))}{\partial (Net_{ki})} \frac{\partial (Net_{ki})}{\partial (b_{km})} \qquad (1.67)$$

Reemplazando la igualdad (1.67) en la ecuación (1.66) se obtiene:

$$\frac{\partial R_i}{\partial b_{km}} = -2(y_{ik} - f_k(x_i))\frac{\partial(g(Net_{ki}))}{\partial(Net_{ki})}\frac{\partial(Net_{ki})}{\partial(b_{km})} \quad (1.68)$$

Reemplazando en esta igualdad las derivadas de las ecuaciones (1.52) y (1.53) se obtiene:

$$\frac{\partial R_i}{\partial b_{km}} = -2(y_{ik} - f_k(x_i))g'(Net_{ki})Z_{mi} = d_{ki}Z_{mi}\ , \quad (1.69)$$

donde

g' representa la derivada de g y $d_{ki} = -2(y_{ik} - f_k(x_i))g'(Net_{ki})$, (1.70)

este valor mide el cambio del error cuando los pesos w varían [9], es por ello que se le llama "sensibilidad del error" en la neurona de la capa de salida.

Este proceso descrito se repite para el total de observaciones del conjunto de entrenamiento (N).

La actualización de los pesos se realiza de la siguiente forma:

$$b_{km}^{(r+1)} = b_{km}^{r} - g\sum_{i=1}^{N}\frac{\partial R_i}{\partial b_{km}}$$

$$b_{km}^{(r+1)} = b_{km}^{r} + g\sum_{i=1}^{N}2(y_{ik} - f_k(x_i))g'(Net_{ki})Z_{mi} \quad (1.71)$$

Actualización de los pesos en la capa Oculta.

La actualización de los pesos de la capa oculta se hace de forma similar que la actualización de los pesos de la capa de salida.

$$\Delta w_{ml} = -g \frac{\partial R}{\partial w_{ml}} = -g \sum_{i=1}^{N} \frac{\partial R_i}{\partial w_{ml}} \tag{1.72}$$

Desarrollando:

$$\frac{\partial R_i}{\partial w_{ml}} = \frac{\partial}{\partial w_{ml}} \left[\sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2 \right] \tag{1.73}$$

$$= -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) \frac{\partial (g(Net_{ki}))}{\partial (w_{ml})}$$

$$\frac{\partial R_i}{\partial w_{ml}} = -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) \frac{\partial (g(Net_{ki}))}{\partial (Net_{ki})} \frac{\partial (Net_{ki})}{\partial (Z_{mi})} \frac{\partial Z_{mi}}{\partial (Net_{mi})} \frac{\partial Net_{mi}}{\partial w_{ml}} \tag{1.74}$$

Tomando las derivadas de las ecuaciones (1.54), (1.52), (1.51), (1.50) y reemplazándolas en la igualdad (1.74), se obtiene la expresión del gradiente del error en la capa oculta:

$$\frac{\partial R_i}{\partial w_{ml}} = -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g'(Net_{ki}) b_{km} s'(Net_{mi}) x_{il} = s_{ml} x_{il} \tag{1.75}$$

Donde $s_{ml} = -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g'(Net_{ki}) b_{km} s'(Net_{mi})$ representa la ''sensibilidad del error'' en la neurona de la capa oculta.

Los pesos de la capa oculta dependen de los pesos de la capa de salida por lo tanto para su actualización, se usa los pesos actualizados de la capa de salida y las N observaciones de entrenamiento.

$$w_{ml}^{(r+1)} = w_{ml}^{r} - g \sum_{i=1}^{N} \frac{\partial R_i}{\partial w_{ml}} \tag{1.76}$$

$$w_{ml}^{(r+1)} = w_{ml}^{r} + 2g \sum_{i=1}^{N} \sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g'(Net_{ki}) b_{km}^{(r+1)} s'(Net_{mi}) x_{il} \qquad (1.77)$$

A este procedimiento de actualización de pesos se denomina Backpropagation o de retro propagación debido a que el error se propaga de manera inversa al funcionamiento normal de la red. De esta forma, el algoritmo encuentra el error en el proceso de aprendizaje desde las capas más internas hasta llegar a la entrada. En base al cálculo de este error se actualizan los pesos. Las funciones de transferencia utilizadas deben ser continuas para que su derivada exista en todo el intervalo, ya que el término $g'(Net_{ki})$ es requerido para el cálculo del error.

Las sensibilidades d_{ki} y s_{ml} están relacionadas mediante la expresión:

$$s_{ml} = -2\sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g'(Net_{ki}) b_{km}^{(r+1)} s'(Net_{mi}) = s'(Net_{mi}) \sum_{k=1}^{K} d_{ki} b_{km}^{(r+1)} \qquad (1.78)$$

1.1.1.3 **Backpropagation para clasificación.**

En esta sección se desarrolla el algoritmo ‘‘backpropagation’’ para la solución del problema de clasificación, la finalidad del uso de este algoritmo en este caso es modelar las probabilidades de salida.

La salida de la red está dada por:

$$f_k(X) = g\left(b_{ok} + b_k^T s\left(w_{om} + w_m^T X\right)\right) = g(Net_k) \qquad (1.79)$$

En este caso se usa una función particular de activación para la capa de entrada , dicha función es la función exponencial definida por:

$$s\left(w_{om}+w_m^T X\right)=s\left(Net_m\right)=1/\left(1+\exp(-Net_m)\right), \tag{1.80}$$

y para la capa oculta la función exponencial normalizada o función de activación softmax (Bridle 1990) está dada por:

$$g(Net_{ki})=\frac{e^{Net_{ki}}}{\sum_{k=1}^{K} e^{Net_{ki}}},\quad Net_k=b_{ok}+b_k^T s\left(w_{om}+w_m^T X\right), \tag{1.81}$$

donde K: es el número de clases.

Bajo estas condiciones las salidas de la red , $f_k(X)$, k=1,2…,K representa una distribución de probabilidades, pues :

$$0\le f_k(X)\le 1 \quad \text{y} \quad \sum_{k=1}^{K} f_k(X)=1 \tag{1.82}$$

Suponiendo que la salida esperada, está dada por:

$$E\left[Y_k/X\right]=f_k(X)=g\left[b_{0k}+b_k^T s\left(w_{0m}+w_m^T X\right)\right], \tag{1.83}$$

para ajustar el modelo se compara la salida real G con la salida de la red $f_k(X)=g(Net_k)$, mediante una función de pérdida:

$$L(G, f_k(X))=-\log P(G), \tag{1.84}$$

donde:

$P(G)$: es la función de probabilidad. Para una red neuronal con una salida G_k para cada clase, la función de probabilidad está dada por:

$$P(G)=\prod_{k=1}^{K}\left(f_k(X)\right)^{G_k} \tag{1.85}$$

$$L(G, f_k(X)) = -\log P(G) = -\log \prod_{k=1}^{K} \left(f_k(X) \right)^{G_K} = -\sum_{k=1}^{K} G_k . \log(f_k(X))$$

$$L(G, f_k(X)) = -\sum_{k=1}^{K} I\left(G = k\right) \log\ f_k(X)\ , \qquad (1.86)$$

A esta función de pérdida se llama "cross entropía" o "deviance".

Para el ajuste del modelo, el objetivo es minimizar la función de error dada por:

$$R = -\sum_{i=1}^{N} \sum_{k=1}^{K} G_{ki} \log\ f_k(x_i) \qquad (1.87)$$

La actualización de los pesos en la capa oculta y en la capa de salida se realiza de manera similar al caso de la regresión.

Para la capa de salida.

Utilizando el método de gradiente descendiente:

$$\Delta b_{km} = -g \frac{\partial R}{\partial b_{km}} = -g \sum_{i=1}^{N} \frac{\partial R_i}{\partial b_{km}} \qquad (1.88)$$

$$\frac{\partial R_i}{\partial b_{km}} = -\frac{\partial R_i}{\partial Net_{ki}} . \frac{\partial Net_{ki}}{\partial b_{km}}$$

$$\frac{\partial R_i}{\partial b_{km}} = -Z_{mi} \sum_{k=1}^{K} \frac{\partial R_i}{\partial f_k(x_i)} . \frac{\partial f_k(x_i)}{\partial Net_{ki}}$$

$$\frac{\partial R_i}{\partial b_{km}} = -\left(G_{ik} - f_k(x_i)\right) Z_{mi} = d_{ki} Z_{mi}\ , \qquad (1.89)$$

Actualizando los pesos de la capa de salida, luego de presentar todas las observaciones de entrenamiento

$$b_{km}^{(r+1)} = b_{km}^{(r)} - g\sum_{i=1}^{N}\frac{\partial R_i}{\partial b_{km}} \tag{1.90}$$

$$b_{km}^{(r+1)} = b_{km}^{(r)} + g\sum_{i=1}^{N}(G_{ik} - f_k(x_i))Z_{mi} \tag{1.91}$$

Si la neurona pertenece a la capa Oculta.

Se tiene:

$$\Delta w_{ml} = -g\frac{\partial R}{\partial w_{ml}} = -g\sum_{i=1}^{N}\frac{\partial R_i}{\partial w_{ml}} \tag{1.92}$$

$$\frac{\partial R_i}{\partial w_{ml}} = -x_{il}\sum_{k=1}^{K}\frac{\partial R_i}{\partial Net_{ki}}.\frac{\partial Net_{ki}}{\partial Net_{mi}}$$

$$\frac{\partial R_i}{\partial w_{ml}} = -x_{il}Z_{mi}(1-Z_{mi})\sum_{k=1}^{K}(G_{ik} - f_k(x_i))b_{km} = s_{ml}x_{il} \tag{1.93}$$

Finalmente los pesos de la capa oculta son actualizados iterativamente mediante la relación:

$$w_{ml}^{(r+1)} = w_{ml}^{(r)} - g\sum_{i=1}^{N}\frac{\partial R_i}{\partial w_{ml}} \tag{1.94}$$

$$w_{ml}^{(r+1)} = w_{ml}^{r} + g\sum_{i=1}^{N}\sum_{k=1}^{K}(G_{ik} - f_k(x_i))b_{km}^{(r+1)}Z_{mi}(1-Z_{mi})x_i \tag{1.95}$$

1.1.2 **Consideraciones sobre el algoritmo de aprendizaje.**

1.1.2.1 **Dimensión de la red neuronal.**

- Respecto al número de capas que se usan para entrenar la red, varios autores recomiendan que tres capas son suficientes (entrada, oculta y salida). Sin embargo, existen problemas que son más fáciles de resolver cuando se tienen más de una capa oculta [13].
- El número de neuronas en la capa de entrada, así como el número de neuronas en la capa de salida suelen ser determinados por la naturaleza de la aplicación; por ejemplo, para problemas de regresión se considera una sola neurona y para problemas de clasificación frecuentemente la cantidad de neuronas es la misma que el numero de categorías. Decidir cuántas neuronas debe tener la capa oculta no suele ser tan evidente, puesto que ésta influye en la eficiencia del aprendizaje y generalización de la red.
- Es posible eliminar neuronas ocultas si la red converge sin problemas, en caso contrario, es posible que sea necesario aumentar este número.

1.1.2.2 **Sobreentrenamiento (*Overfitting)***

En general, cuando la red neuronal presenta un número grande de capas y neuronas, el número de parámetros a ajustar durante el entrenamiento es muy elevado, puede que esta red aún siendo capaz de aprender los ejemplos de entrenamiento, no sea capaz de **generalizar**. Este problema recibe el nombre de **sobreentrenamiento** (``*overfitting*''), y refleja el comportamiento existente entre el sesgo y la varianza [13],[4], tal como se muestra en el apéndice.

1.1.3 **Técnicas de control de sesgo y varianza.**

1.1.3.1 **Arquitectura de la red.**

Para resolver un problema particular, primero se define la arquitectura de la red; esto es, el número de neuronas en la capa de entrada, oculta y salida. El número de neuronas en la(s) capa(s) oculta juega un papel importante en la definición de la red. Se dice que una red es compleja si presenta un mayor número de neuronas en la(s) capa(s) oculta(s) con respecto a una red simple que tiene pocas neuronas.

El control del grado de complejidad de la red consiste en el incremento o disminución del número de neurona(s) en la(s) capa(s) ocultas, con el incremento de neuronas el número de parámetros del modelo aumenta. El aprendizaje tiende a obtener una solución basada más en la interpolación que en la aproximación, es decir, la función obtenida tiende a ajustarse más a los puntos de entrenamiento, por ello la varianza aumenta (figura 1.14) ya que la solución obtenida depende más del conjunto de entrenamiento [13]. No obstante, el sesgo se reduce ya que existen más grados de libertad para obtener una solución. En el caso límite, cuando la función es una interpolación exacta, como se ilustra en la figura 1.13 (b), el sesgo sería mínimo y la varianza sería máxima ya que esta dependería completamente del conjunto de datos de entrenamiento.

Otra medida de control del sesgo y varianza es el número de iteraciones en el proceso de aprendizaje de la red, con la disminución del número de iteraciones se tiene un incremento en el sesgo, y con el aumento del número de iteraciones la varianza tiende a aumentar (figura 1.12). Esto se debe a que la

(2.16)

red con mayor número de iteraciones se ajusta más a las observaciones de entrenamiento.

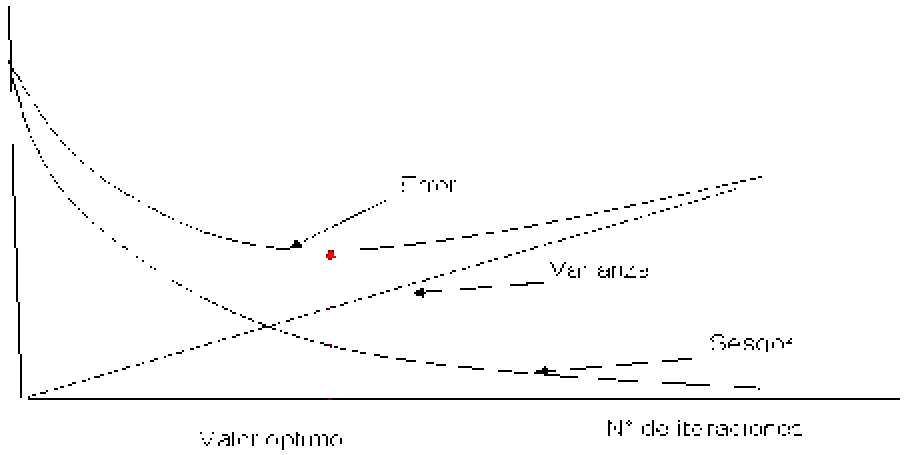


Figura 1.12: Comportamiento de la varianza y el sesgo con respecto al numero de iteraciones.

La fuente de error entre una red neuronal con menor grado de complejidad (figura 1.13a) y otro con mayor grado (figura 1.13b) es diferente. En el primer caso, está asociado mas con el sesgo; en cambio en el segundo caso, el error es atribuido en gran parte a la variabilidad del conjunto de entrenamiento.

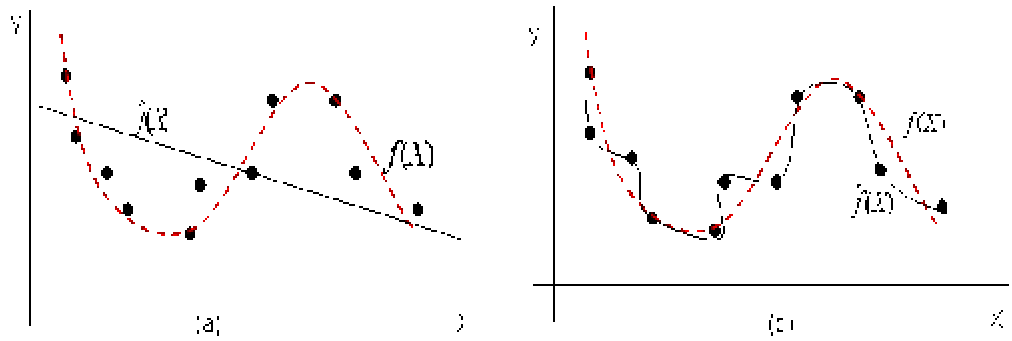


Figura 1.13: Esquema que ilustra los conceptos de sesgo y varianza.

Los círculos simbolizan un conjunto de puntos generados a partir de la función $f(X)$ (línea discontinua) mediante la adición de un error. El objetivo es aproximar $f(X)$ tanto como sea posible. En (a) intentamos modelar el sistema

mediante un polinomio de grado 1, esto es una red con una neurona en la capa oculta y nos da una pobre representación de $f(X)$. Podemos tener mejor ajuste incrementando el orden del polinomio y esto se logra con el incremento de neuronas en la capa oculta y como consecuencia se tiene mayor número de parámetros y de grados de libertad como se observa en (b), el modelo en este caso se forma por una interpolación exacta de los datos, en este caso el modelo se centra en la varianza de los datos y el sesgo es mínimo.

1.1.1.1 **Combinación de las predicciones de la Red Neuronal.**

Otro procedimiento de control es la combinación lineal de las predicciones[15], la idea principal es generar por algún medio un conjunto de predicciones usando un algoritmo de aprendizaje y combinarlos por algún mecanismo, esta técnica será abordada con detalle en el siguiente capitulo.

1.2 Selección y Evaluación de Modelos.

Para la selección y evaluación de modelos de red neuronal, se divide la base de datos en dos partes; uno para entrenar la red (ajustar el modelo) y otro para la prueba.

Con el modelo ajustado en base a los datos de entrenamiento se obtiene el error de predicción.

El objetivo que tenemos es seleccionar un modelo de la red neuronal que tenga mejor desempeño en la generalización (mejores predicciones), es decir que presente menor error promedio para el conjunto de nuevas observaciones (error prueba) no necesariamente que tenga mínimo error promedio para el conjunto de observaciones de entrenamiento (error de entrenamiento).

El error de entrenamiento frecuentemente puede ser menor que el error de prueba (figura 1.14), puesto que los datos de entrenamiento son inicialmente

usados para el ajuste y evaluación del error. Un modelo ajustado frecuentemente se adapta a los datos de entrenamiento. El error de entrenamiento decrece a medida que el modelo tiene mayor grado de complejidad, mayor número de neuronas en la capa oculta, en este caso la red presenta mayor número de parámetros y grados de libertad y esto evidentemente se relaciona con un incremento en la varianza y una reducción del sesgo, debido a que el modelo se ajustará más al conjunto de observaciones de entrenamiento; sin embargo, la generalización de estos modelos, predicción de nuevas observaciones, no siempre es muy buena.

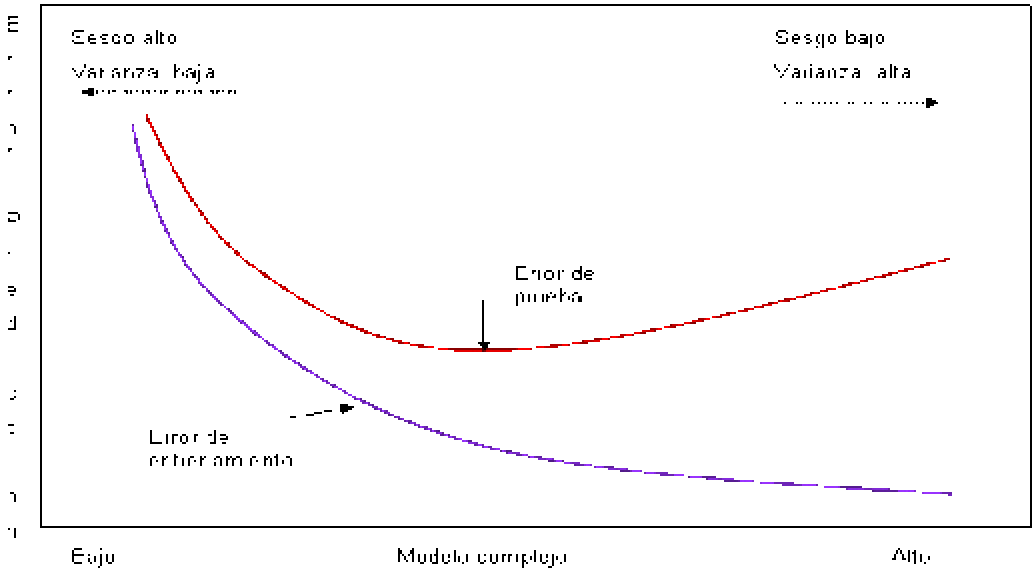


Figura 1.14: Comportamiento del error de prueba y de entrenamiento, en un modelo en el cual el grado de complejidad varía.

1.2.1 **Error de entrenamiento y de prueba**

Considerando una variable respuesta Y, relacionada con un vector de entradas X mediante $Y = f(X) + e$, y una estimación de $f(X)$, $\hat{f}(X)$, se obtiene a partir del conjunto de observaciones de entrenamiento; la discrepancia que existe entre Y y $\hat{f}(X)$, se mide con una función de pérdida $L(Y, \hat{f}(X))$, llamada error de predicción.

Si Y es numérica, L se define como:

$$L(Y, \hat{f}(X)) = (Y - \hat{f}(X))^2 \qquad (1.96)$$

La esperanza de esta función, $E_{rr} = E\{L(Y, \hat{f}(X))\}$, sobre las observaciones de prueba , se llama error de prueba [9].

Por otro lado, el error de entrenamiento [9], se define como el promedio del error de predicción sobre las observaciones de entrenamiento, esto es,

$$\overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)) = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2 \qquad (1.97)$$

Si la respuesta es categórica $G \varepsilon C = \{1,2,.., K\}$ donde los números son etiquetas que representan clases $\{G_1, G_2,.., G_K\}$, la función de pérdida comúnmente usada para la clasificación es definida por:

$$L(G, \hat{G}(X)) = I(G \neq \hat{G}(X)) = \begin{cases} 1 & G \neq \hat{G}(X) \\ 0 & G = \hat{G}(X) \end{cases}, \qquad (1.98)$$

donde $\hat{G}(X) = \arg\max_k \hat{p}_k(X)$

Para este caso, el error de prueba se define mediante la siguiente relación:

$$Err = E\left(L\left[G, \hat{G}(X)\right]\right) \qquad (1.99)$$

y el error de entrenamiento por:

$$\overline{err} = \frac{\sum_{i=1}^{N} I(G \neq \hat{G}(x_i))}{N} \qquad (1.100)$$

También se puede utilizar como función de pérdida para densidades de la familia exponencial, la entropía

$$L(G, p(X)) = -\sum_{k=1}^{K} I(G = k) \log p_k(X) \qquad (1.101)$$

El error de prueba para esta medida se define como:

$$E_{rr} = E\{L(G, \hat{G}(X))\} \qquad (1.102)$$

y el error de entrenamiento

$$\overline{err} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} G_{ik} \log p(x_i) \qquad (1.103)$$

El error de entrenamiento es una estimación excesivamente optimista del error de prueba puesto que el error de entrenamiento frecuentemente es menor que el error de prueba.

1.2.2 **Validación cruzada (Cross-Validation)**

Es el método más simple para estimar el error de prueba, para tal efecto se divide al azar el conjunto de datos en T subconjuntos de datos generalmente T varia entre 5 y 10, con los T-1 subconjuntos se entrena la red y el otro subconjunto sirve para la prueba, por ejemplo para T=5 se tiene:

1	2	***3***	4	5
Entrenamiento	Entrenamiento	***Prueba***	Entrenamiento	Entrenamiento

Denotemos con $\hat{f}^{-t}(x)$, función ajustada con el t-ésimo subconjunto de entrenamiento, la validación cruzada estima el error de prueba mediante:

$$E_{rr(cv_t)} = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}^{-t(i)}(x_i)\right) \quad \text{t=1,..T} \qquad (1.104)$$

Desde que nuestro objetivo es encontrar una red que tenga mejor desempeño para un nuevo conjunto de datos, la forma simple de comparar varias redes, es evaluando alguna función de error para el conjunto de prueba y seleccionando aquella que tenga menor error de prueba.

1.2.3 **Método Bootstrap.**

El Bootstrap (Efron 1979) constituye la técnica más versátil y conocida dentro del método de remuestreo. La idea básica es tratar la(s) muestra(s) como si fuera la población, (debido a la analogía entre muestra y población) y a partir de ella extraer con reposición un gran número de remuestras del mismo tamaño de este. Cada remuestra será muy probablemente, algo diferente de la muestra original, con lo cual un estadístico $\hat{q}^*$, calculado a partir de una de esas remuestras tomara un valor diferente del que produce otra remuestra y del $\hat{q}$ observado. La afirmación fundamental del bootstrap es que una distribución de frecuencia de estos $\hat{q}^*$ calculados a partir de estas remuestras es una estimación de la distribución muestral de $\hat{q}$ (Monney y Duval, 1993)

El bootstrap tiene la ventaja sobre los métodos tradicionales de no requerir formulaciones teóricas y poder emplearse para cualquier estimador por complejo que este sea. Es decir permite efectuar inferencias estadísticas sin necesidad de postular previamente que la distribución cumpla ciertas hipótesis.

Una forma de estimar el error de prueba es aplicando bootstrap, para ello se selecciona T replicas de bootstrap (T conjuntos de entrenamiento usualmente T=100). Para cada conjunto de datos se ajusta una red. La estimación del error de prueba esta dado por:

$$\hat{Err}_{Boot} = \frac{1}{T}.\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N} L\left(y_i, \hat{f}^{*t(i)}(x_i)\right) \qquad (1.105)$$

$\hat{f}^{*t}(x_i)$ es la predicción realizada por la t-ésima red para una observación x_i de entrenamiento.

CAPÍTULO II: COMBINACIÓN DE CLASIFICADORES. MÉTODOS BAGGING, BOOSTING.

2.1 Introducción.

La forma clásica de seleccionar un modelo consiste en entrenar varios y, posteriormente, seleccionar aquél que ofrezca un mejor comportamiento en un conjunto de observaciones no empleadas en su ajuste, pero bajo este criterio se pierde la información que proporcionan los modelos con menores índices globales. Una forma de evitar este tipo de problemas consiste en combinar los modelos.

En redes neuronales uno de los campos actuales de investigación, se centra en el modo en que se pueden combinar, unir o ensamblar predicciones de clasificadores obtenidos de varias redes neuronales con la finalidad de obtener uno o más modelos que sean más eficientes; esto es, un clasificador con mejores resultados desde el punto de vista ''clasificatorio''. Al clasificador que se obtiene de esta unión se le da el nombre de combinación de clasificadores (multiclasificador o comité)[15].

Una combinación de clasificadores puede mejorar el resultado de los clasificadores individuales que lo componen únicamente si existen desacuerdos entre ellos; esto es, si existen casos a los que distintos clasificadores asignan distinta clasificación (Hand 1997) [15]. Se busca, por lo tanto, diversidad en los clasificadores que serán componentes de una combinación.

Para aclarar el concepto anterior supongamos que tenemos tres clasificadores: $\{G_1(x), G_2(x), G_3(x)\}$ y consideremos un nuevo caso x_{N+1}.

Podemos tener varias posibilidades de clasificación para este caso:

1) La clasificación realizada por cada uno de los tres clasificadores es incorrecta.
2) La clasificación realizada por cada uno de los tres clasificadores es correcta.
3) Dos clasificadores predicen incorrectamente y el otro realiza una predicción correcta.
4) Dos clasificadores predicen correctamente y el otro realiza una predicción errónea.

En el primer y segundo caso, una ‘’combinación’’ de clasificadores no variaría el resultado final, mientras para el tercero y cuarto caso existen varias alternativas de combinación que podrían hacer variar los resultados de cada combinación. Más adelante se desarrollarán algunos métodos de combinación.

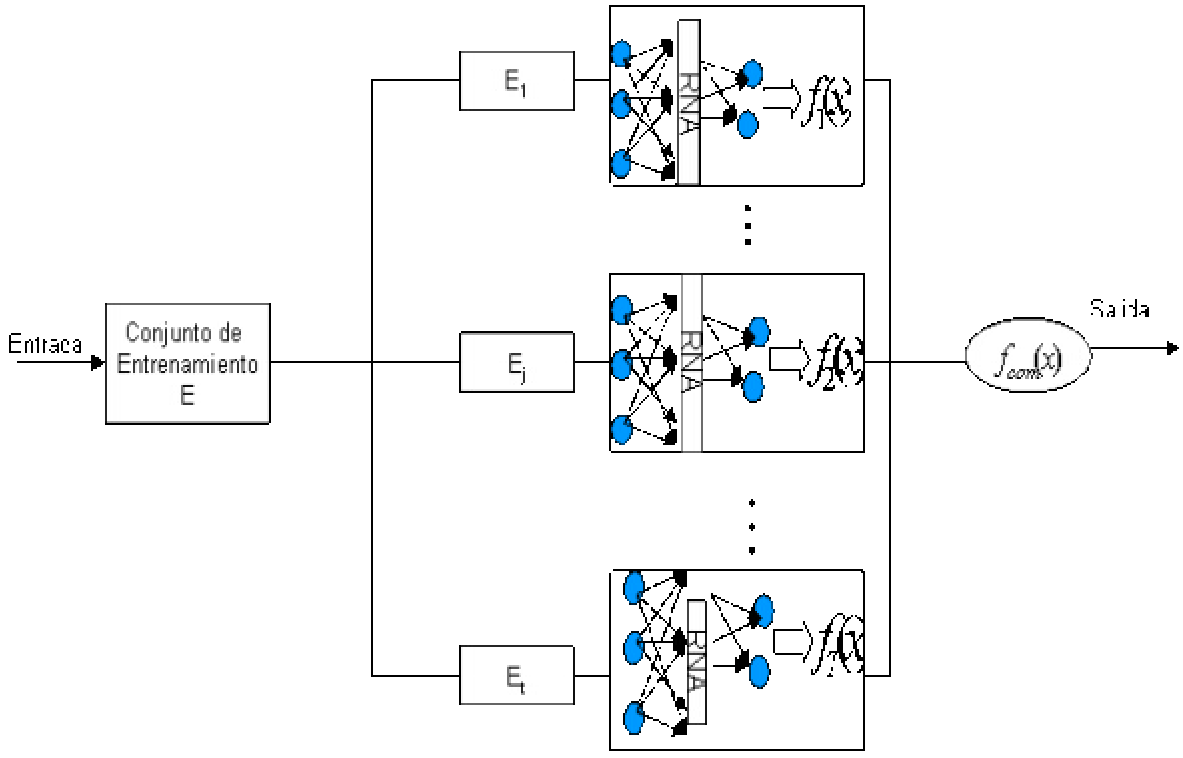


Figura 2.1: Esquema de una combinación de predicciones de clasificadores para redes neuronales [17].

En la figura 2.1 se ilustra un esquema de una combinación de clasificadores. La idea básica de la combinación de clasificadores ,$f_{com}(x)$, es la construcción de un conjunto de clasificadores base, $f_1(x),\ldots,f_T(x)$, cada uno de los cuales es obtenido usando distintos conjuntos de entrenamiento, en la práctica estos conjuntos se obtienen de manera artificial. La técnica mas usada es el re-muestreo.

2.2 Combinación de clasificadores.

El elemento vital de la combinación de clasificadores es el problema de inestabilidad del clasificador base. Entendiéndose como clasificador inestable aquél que tiene fuerte dependencia del conjunto de entrenamiento [6], estos clasificadores se caracterizan por generar grandes cambios: en la predicción, en la varianza y en el error cuando se realizan pequeños cambios en el conjunto de entrenamiento. Otra característica importante de los clasificadores inestables es que poseen *pequeño sesgo*, en un conjunto de datos. El problema radica en la *varianza,* lo cual puede reducirse combinando diferentes clasificadores base generados a partir de modificaciones del conjunto de entrenamiento.

A continuación definimos la combinación de clasificadores para problemas de regresión y clasificación.

2.2.1 Caso de la Regresión.

Consideremos los conjuntos de entrenamiento E_1,.., E_T de una red neuronal obtenidos a partir del conjunto de entrenamiento inicial E = { (x_i, y_i) / $x_i \varepsilon \square^p$, $y_i \varepsilon \square$; i=1,…N }, por métodos que después aclararemos.

Para cada uno de los conjuntos de entrenamiento E_1,.., E_T se ajusta un modelo de regresión $f_j(x)$ con la finalidad de aproximar $f(x)$.

La forma de combinar predicciones de clasificadores [1], [5],[9] está dada por una combinación lineal de los $f_j(x)$:

$$f_{com}(x) = \sum_{j=1}^{T} c_j . f_j(x) \quad \text{con } c_j > 0 \text{ y } \sum_{j=1}^{T} c_j = 1 \qquad (2.1)$$

Donde:

$f_j(X)$: Es la predicción del j-ésimo miembro de la combinación.

c_j : son coeficientes que pondera la importancia de cada miembro $f_j(X)$.

$f_{com}(x)$: es la combinación de predicciones de clasificadores.

T es el número de redes neuronales entrenadas.

Un caso particular se obtiene cuando los $c_j = \frac{1}{T}$, $\forall j$ (Toda las $f_j(X)$ tienen la misma importancia).

2.2.2 Caso de la Clasificación.

Consideremos los conjuntos de entrenamiento E_1,.., E_T obtenidas en base al conjunto de entrenamiento inicial, E={ (x_i, G_i)/ $x_i \varepsilon$ □ ^{p}, $G_i \varepsilon C$ ={1,2,.., K} ;i=1,..N }.

En cada uno de los conjuntos de entrenamiento E_j se entrena la red para resolver el mismo problema con el objetivo de obtener un clasificador $G_j(x)$.

Al final tenemos T clasificadores base ,$G_j(x)$, j=1,…,T cada uno de los cuales asigna a la observación x una etiqueta $g \in C = \{1,2,..,K\}$, esto es $G_j(x) = g$. La relación $G_1(x) = G_2(x) = ... = G_T(x)$, no siempre se cumple, por tanto puede existir discrepancias entre los clasificadores base; una regla natural en la realidad para resolver estas discrepancias es el principio del voto mayoritario, esto es si la mayoría,(más de la mitad), de los T clasificadores votan por la misma etiqueta entonces el clasificador combinado toma esa etiqueta como resultado final, en caso de empate se decide al azar.

La combinación de predicciones de clasificadores base , $G_j(x)$, en problemas de clasificación está dada por el principio del voto mayoritario [1], [5],[9].

$$G_{com}(x) = \arg\max_{g \in C=\{1,2,..K\}} \sum_{j=1}^{T} I\left(G_j(x) = g\right). \qquad (2.2)$$

Esta combinación, $G_{com}(x)$, asigna a la observación x, la etiqueta g del argumento de $I(\bullet)$ que hace máximo el número de clasificadores base que predicen esa etiqueta , $\sum_{j=1}^{T} I\left(G_j(x) = g\right)$.

La definición (2.2) se puede expresar equivalentemente mediante:

$$G_{com}(x) = k \text{ , sii } \quad N(k) = \max_{g \in C=\{1,2,..K\}} N(g) \text{ , donde } N(g) = \sum_{j=1}^{T} I\left(G_j(x) = g\right) \qquad (2.3)$$

2.3 Error de la combinación de clasificadores.

Los clasificadores base deben presentar diversidad en los resultados como una condición para reducir el error de la combinación de clasificadores, esto es clasificadores independientes y que hagan que los errores no estén correlacionados, obviamente clasificadores base similares en rendimiento no ayudan en mejorar la precisión [15].

La motivación de la combinación de clasificadores puede entenderse analizando el error para la combinación de clasificadores, lo que es particularmente simple si consideramos el problema de regresión.

Supongamos que tenemos T modelos entrenados por la red neuronal, $f_j(x)$ j=1,..,T y representemos por $f(x)$ la función a aproximar. Cada uno de los modelos se puede expresar mediante:

$$f_j(x) = f(x) + e_j \text{ ,} \qquad (2.4)$$

siendo que e_j es el error cometido por la j-ésima red neuronal ante la observación x . El error cuadrático para el modelo $f_j(x)$ es:

$$E_j = E\left[\left\{f_j(x) - f(x)\right\}^2\right] = E\left[e_j^2\right], \tag{2.5}$$

y el error cuadrático promedio , que representa cada modelo es:

$$E_{medio} = \frac{1}{T}\sum_{j=1}^{T} E_j = \frac{1}{T}\sum_{j=1}^{T} E\left[e_j^2\right] \tag{2.6}$$

Considerando una combinación de modelos, cuya salida es la media de todas las respuestas de los distintos modelos, es decir, todas las redes individuales tienen la misma importancia $c_j = 1/T$.

$$f_{com}(x) = \frac{1}{T}\sum_{j=1}^{T} f_j(x) \tag{2.7}$$

reemplazando (2.4) en (2.7) se obtiene:

$$f_{com}(x) = \frac{1}{T}\sum_{j=1}^{T}\left(f(x) + e_j\right) = f(x) + \frac{1}{T}\sum_{j=1}^{T} e_j \quad , \tag{2.8}$$

el error de la combinación está dado por:

$$Err_W = E\left[\left(\frac{1}{T}\sum_{j=1}^{T} f_j(x) - f(x)\right)^2\right] = E\left[\left(\frac{1}{T}\sum_{j=1}^{T} e_j\right)^2\right] \tag{2.9}$$

Asumiendo que los errores tienen media cero y que no están correlacionados $E\left[e_j\right] = 0$, $E\left[e_j e_i\right] = 0$ si $i \neq j$,el error de la combinación está dado por:

$$Err_W = \frac{1}{T^2} E\left[\left(\sum_{j=1}^{T} e_j\right)^2\right] = \frac{1}{T^2}\sum_{j=1}^{T} E\left(e_j^2\right) = \frac{1}{T}.E_{medio} \leq E_{medio} \tag{2.10}$$

De este resultado, el error se puede reducir fácilmente combinando el mayor número de modelos posibles. En la práctica no se cumple ya que la mayoría de las salidas de los modelos están correlacionadas y la reducción del error es

más pequeña. Utilizando la desigualdad de Cauchy $E\left[\left(\sum_{j=1}^{T} e_j\right)^2\right] \le \left[T\sum_{j=1}^{T} E\left(e_j\right)^2\right]$ se tiene:

$$Err_W = E\left[\left(\frac{1}{T}\sum_{j=1}^{T} e_j\right)^2\right] = \frac{1}{T^2}E\left[\left(\sum_{j=1}^{T} e_j\right)^2\right] \le \frac{1}{T^2}\left[T\sum_{j=1}^{T} E\left(e_j\right)^2\right] \le \frac{1}{T}\sum_{j=1}^{T} E\left(e_j\right)^2 = E_{medio} \quad (2.11)$$

Aunque las soluciones anteriores (2.10) y (2.11) proporciona cierta mejora de los resultados de manera más sencilla, no estamos considerando la combinación óptima de los modelos, ya que todos asignan la misma importancia en la decisión final.

Ahora consideramos la combinación general:

$$f_{com}(x) = \sum_{j=1}^{T} c_j f_j(x) \quad \text{con} \quad \sum_{j=1}^{T} c_j = 1 \quad (2.12)$$

Reemplazando (2.4) en (2.12) se tiene:

$$f_{com}(x) = \sum_{j=1}^{T} c_j f_j(x) = f(x) + \sum_{j=1}^{T} c_j e_j \quad (2.13)$$

El error de la combinación para este caso resulta ser:

$$Err_{com} = E\left[\left(f_{com}(x) - f(x)\right)^2\right] = E\left[\left(f(x) + \sum_{j=1}^{T} c_j e_j - f(x)\right)^2\right] \quad (2.14)$$

$$Err_{com} = E\left[\left(\sum_{j=1}^{T} c_j e_j\right)^2\right] = E\left[\left(ce^T\right)^2\right] = E\left[ce^T ec^T\right] = cE\left(e^T e\right)c^T = c.C.c^t \quad (2.15)$$

donde : $\vec{c} = (c_1,...,c_T)$, $\vec{e} = (e_1,...,e_T)$ y C es la matriz de covarianza de los errores cuyos componentes son:

$$C_{ij} = E\left[e_i(x).e_j(x)\right] \quad (2.16)$$

Los valores óptimos de los coeficientes se obtienen minimizando $Err_{com}(x)$. Para esto usamos los multiplicadores de Lagrange l, con la restricción $\sum_{j=1}^{T} c_j = 1$

$$\frac{\partial}{\partial c^T}\left[\vec{c}.C.\vec{c}^{\,t} - l\left(\vec{1}.\vec{c}^{\,t} - 1\right)\right] = 0 \ , \ \vec{1} = (1,..,1)_{1xT} \ , \qquad (2.17)$$

de donde, el mínimo de la ecuación (2.15) ocurre cuando

$$2.C.\vec{c}^{\,t} - l.\vec{1}^{\,t} = 0 \qquad (2.18)$$

Los clasificadores base son linealmente independientes si estos presentan diversidad en las predicciones. Cabe recordar que para reducir el error de la combinación, uno de los requisitos es que los clasificadores base presenten diversidad en los resultados.

La matriz de covarianza presenta inversa desde que los clasificadores son linealmente independientes.

$$\vec{c}^{\,t} = \frac{l}{2}.C_{TxT}^{-1}.\vec{1}_{Tx1}^{\,t} \qquad (2.19)$$

Desde que la matriz C es simétrica, los valores de los coeficientes se obtienen mediante:

$$\vec{c} = \frac{l}{2}.\vec{1}_{Tx1}.C_{TxT}^{-1} \qquad (2.20)$$

Sustituyendo $\vec{c}^{\,t} = \frac{l}{2}.C_{TxT}^{-1}.\vec{1}_{Tx1}^{\,t}$ en $\sum_{i=1}^{T} c_i = \vec{1}.\vec{c}^{\,t} = 1$, se tiene:

$$\vec{1}.\frac{l}{2}.C_{TxT}^{-1}.\vec{1}_{Tx1}^{\,t} = 1 \qquad (2.21)$$

$$l = \frac{2}{\vec{1}.C^{-1}.\vec{1}^{\,t}} \qquad (2.22)$$

Reemplazando este resultado en la ecuación (2.20) se tiene:

$$\vec{c} = \frac{\vec{1}_{Tx1}.C_{TxT}^{-1}}{\vec{1}.C^{-1}.\vec{1}^{\,t}}. \qquad (2.23)$$

$$c_i = \frac{\sum_{j=1}^{T} a_{ij}}{\sum_{k=1}^{T}\sum_{j=1}^{T} a_{kj}} \quad , \quad i = 1,...,T \tag{2.24}$$

Si los errores $e_i(x)$ no están correlacionados y tienen media cero, $C_{ij} = 0, \forall i \neq j$, el valor óptimo de c_i, tiene la forma :

$$c_i = \frac{s_i^{-2}}{\sum_{j=1}^{T}\left[s_j^{-2}\right]}, \text{ donde } s_i^2 = E(e_i^2) \tag{2.25}$$

Finalmente reemplazando (2.23) en (2.15) se tiene la estimación del error de la combinación, mediante:

$$Err_{com} = \vec{c}.C.\vec{c}^{\,t} = \frac{\vec{1}_{Tx1} C^{-1}_{TxT}}{\vec{1}.C^{-1}.\vec{1}^{\,t}}.C.\frac{C^{-1}_{TxT}.\vec{1}^{\,t}_{Tx1}}{\vec{1}.C^{-1}.\vec{1}^{\,t}} = \frac{1}{\vec{1}.C^{-1}.\vec{1}^{\,t}} \tag{2.26}$$

Desarrollando se tiene:

$$Err_{com} = \left(\sum_{i=1}^{T}\sum_{j=1}^{T} a_{ij}\right)^{-1} \tag{2.27}$$

El error para la combinación general , $f_{com}(x) = \sum_{j=1}^{T} c_j f_j(x)$, es $Err_{com} = \vec{c}.C.\vec{c}^{\,t} = \left[\vec{1}.C^{-1}.\vec{1}^{\,t}\right]^{-1}$; donde $\vec{c}$ está dado por (2.23) y el error de la combinación para el caso particular , $c_j = 1/T \quad \forall j$, esto es ,$\vec{c} = \frac{1}{T}.\vec{1}$, está dado por (2.9), $Err_W = \frac{1}{T^2}\vec{1}.C.\vec{1}^{\,t}$.

Desde que (2.7) es un caso particular de (2.12), se cumple:

$$Err_{com} \leq Err_W \tag{2.28}$$

Observaciones.

El error de la combinación general puede descomponerse en la suma de dos términos (Krogh y Vedelsby, 1995)

$$E\left[\left(f_{com}(x)-f(x)\right)^2\right]=\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f(x)\right)^2\right]-\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f_{com}(x)\right)^2\right] \qquad (2.29)$$

Para verificar esta relación, desarrollemos el primer término del segundo miembro.

$$\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f(x)\right)^2\right]=\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f_{com}(x)+f_{com}(x)-f(x)\right)^2\right] \qquad (2.30)$$

$$\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f(x)\right)^2\right]=\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f_{com}(x)\right)^2+2\left(f_j(x)-f_{com}(x)\right)\left(f_{com}(x)-f(x)\right)+\left(f_{com}(x)-f(x)\right)^2\right]$$

$$\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f(x)\right)^2\right]=\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f_{com}(x)\right)^2\right]+E\left[\left(f_{com}(x)-f(x)\right)^2\right]$$

Finalmente, se tiene:

$$E\left[\left(f_{com}(x)-f(x)\right)^2\right]=\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f(x)\right)^2\right]-\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f_{com}(x)\right)^2\right] \qquad (2.31)$$

El primer término del lado derecho de la descomposición, $\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f(x)\right)^2\right]=\sum_{j=1}^{T}c_j.E_j$, representa el error promedio ponderado de las funciones base, al segundo termino , $\sum_{j=1}^{T}c_j.E\left[\left(f_j(x)-f_{com}(x)\right)^2\right]$, se le denomina ambigüedad de la combinación y contiene toda las correlaciones entre redes base. Mientras mayor sea la ambigüedad mayor será la reducción del error de la combinación.

Si nosotros tenemos un conjunto de modelos base, el error de la combinación resulta menor o igual al error promedio ponderado, $Err_{com} = \sum_{j=1}^{T} c_j . E$. Particular si $c_j = 1/T \quad \forall j$, se cumple $Err_{com} \leq E_{medio}$.

Finalmente, de los resultados anteriores se cumple la relación:

$$Err_{com} \leq Err_W \leq E_{medio} \qquad (2.32)$$

De este resultado se concluye que la combinación de predicciones de clasificadores reduce el error y por tanto reduce la varianza.

2.1 Métodos de combinaciones de clasificadores.

En la actualidad existen varios métodos de combinación de clasificadores; Bagging (Breiman 1996) [5], Boosting (Shapire 1990, Freud & Shapire 1997) [10] y otros. Ambos métodos construyen un conjunto de clasificadores base $G_j(x)$ y en base a ellos se forma un nuevo clasificador llamado ‘’combinación de clasificadores’’.

Para construir el conjunto de clasificadores base para los métodos , Bagging y Boosting se debe repetir la construcción del clasificador base inicial sobre distintos conjuntos de entrenamiento obtenidas por medio del muestreo a partir de E.

Bagging y Boosting se diferencian en la forma de obtener los conjuntos de entrenamiento y en la forma de combinar los clasificadores base.

2.1.1 Método Bagging (Bootstrap aggregating).

Bagging [5] es un método que genera múltiples muestras, $E_{1,\ldots,} E_T$ paralelamente, a partir del conjunto de entrenamiento inicial, E= {(x_i, y_i) / x_i=(x_{i1},, x_{ip}) i=1,…N }, utilizando muestreo aleatorio simple con reemplazo. A este tipo de muestras se le denomina muestras bootstrap.

Cada muestra bootstrap, E_j, tiene aproximadamente 63.2% de observaciones del conjunto de entrenamiento inicial, E cuando N es suficientemente grande:

$$P(x_i \in E_j) = 1 - \left(1 - \frac{1}{N}\right)^N \approx 1 - (e)^{-1} \approx 0.632 \qquad (2.33)$$

En cada conjunto de entrenamiento, E_j, se entrena la red neuronal con la finalidad de construir un clasificador base $f_j(x)$ j=1,…,T. Los T clasificadores obtenidos son combinados por medio del principio del voto mayoritario o por el promedio. Al resultado de esta combinación se le denomina ‘’clasificador bagging’’.

A continuación trataremos formalmente el concepto del clasificador bagging para problemas de regresión y clasificación.

Un clasificador bagging para problemas de regresión es:

$$f_{com}(x) = \frac{1}{T}\sum_{j=1}^{T} f_j(x) \qquad (2.34)$$

Este modelo es un caso particular de la definición 2.1 y se obtiene cuando toda las redes base presentan la misma importancia $c_j = \frac{1}{T}$.

Para problemas de clasificación, el ‘’clasificador bagging’’ es un caso particular de la definición 2.2. Usando T clasificadores $,G_j(x)$, se construye la combinación de clasificadores bagging $,G_{com}(x)$, mediante el principio de voto mayoritario ,esto es:

$$G_{com}(x) = \underset{g \in C = \{1,2,..,K\}}{\arg\max} \sum_{j=1}^{T} I\left(G_j(x) = g\right). \qquad (2.35)$$

El algoritmo de "bagging'' que presentó Breiman [5], para obtener el clasificador es un resumen de lo anterior.

……

Algoritmo 2.1: Bagging Breiman(1996)

1.- Dada una muestra de entrenamiento E, un clasificador G(x) , T muestras bootstrap.

2.- Desde j=1,.., T {

3.- E_j =muestra bootstrap proveniente de E.

4.- Para cada muestra E_j, entrenamos la red y tenemos un clasificador $G_j(x)$ o $f_j(x)$ }

5.-Salida:

a) Regresión $f_{com}(x) = \frac{1}{T}\sum_{j=1}^{T} f_j(x)$

$$G_{com}(x) = \arg\max_{G \in C} \sum_{j=1}^{T} I\left(G_j(x) = G\right) \quad C = \{1,2,..,K\}$$

b) Clasificación

6.- Salida de la combinación de clasificadores Bagging.

……

2.1.2 Método Boosting

Con el método de "boosting'', a diferencia de bagging, los clasificadores base no se obtienen paralelamente ya que los pesos de las observaciones se

actualizan en forma secuencial de acuerdo a los resultados obtenidos por el clasificador previo.

El proceso que describe el método se indica a continuación.

Los clasificadores base son construidos en forma secuencial a partir de conjuntos de entrenamiento, $E_j = \{ (v_i x_i, G_i) / v_i x_i = (v_i x_{i1}, \ldots, v_i x_{ip}), i=1,..N \}$, donde: $(x_i, G_i) \varepsilon E$.

Para construir el primer clasificador base $G_1(x)$ se asume que las observaciones de E_1 son ponderadas con pesos iguales v_i. En base a $G_1(x)$ se actualizan los pesos de las observaciones de E_2, de manera que las observaciones que fueron erróneamente clasificadas por G_1 presentan un incremento en sus pesos, mientras que observaciones correctamente clasificadas por G_1 presentan una reducción en sus pesos. Con E_2, se construye $G_2(x)$ y éste sirve para actualizar los pesos del conjunto de entrenamiento, E_3. Este proceso se repite sucesivamente hasta obtener los T clasificadores base.

El clasificador boosting, $G_{com}(x)$ se forma combinando los clasificadores base usando votación simple [10].

Existen dos formas mejoradas de Boosting: Adaboosting (Freund & Schapire 1996) y Arcing (Breiman 1996).

2.1.2.1 Adaboosting para 2 clases.

El método adaboosting se forma usando ‘’votación ponderada’’:

$$G(x) = sign\left(\sum_{j=1}^{T} c_j G_j(x)\right) \qquad (2.36)$$

Donde $G_j(x)$ son predicciones de clasificadores base que produce $\{-1,1\}$ y c_j representan los coeficientes (contribuciones de cada clasificador base $G_j(x)$).

El coeficiente, c_j pondera la importancia de cada clasificador base y su valor depende de su eficiencia en el conjunto de entrenamiento que se usó para construirlo. Si el j-ésimo error de entrenamiento ponderado, err_j es pequeño, el clasificador base $G_j(x)$ adquiere mayor importancia; en caso contrario presenta, menor influencia en la combinación de clasificadores.

El método ‘’adaboosting’’ como un caso particular del método boosting genera las muestras de entrenamiento en forma secuencial, la diferencia es que el clasificador combinado ‘’adaboosting’’ se obtiene por votación ponderada.

El proceso anterior se resume en el siguiente algoritmo de adaboosting [10] para 2 clases:

Algoritmo 2.2: Adaboosting para 2 clases

1.- Ponderar las observaciones con pesos $v_i = 1/N$,i=1,…,N

2.- Repetir j=1 hasta T

a) Ajustamos una red para obtener un clasificador $G_j(x) \in \{-1,1\}$ en base al conjunto de entrenamiento modificado por los coeficientes v_i

b) Calculamos $err_j = \dfrac{\sum_{i=1}^{N} v_i I(G_i \neq G_j(x))}{\sum_{i=1}^{N} v_i}$

c) Calculamos los coeficientes $c_j = \log\left(\dfrac{1-err_j}{err_j}\right)$

d) Los pesos se modifican mediante la relación:

$v_i \leftarrow v_i . \exp\left[c_j . I\left(G_i \neq G_j(x_i)\right)\right]$ i=1,.., N

3.- La predicción de la combinación de clasificadores es:

$$G_{com}(x) = sign\left(\sum_{j=1}^{T} c_j G_j(x)\right)$$

……

Freud & Shapire (1996), Breiman (1998) y Shapire & Singer (1998) han propuesto una generalización de adaboosting al que denominaron "real adaboosting" [8]. En esta generalización los clasificadores base se obtienen mediante la relación: $G_j(X) \leftarrow \dfrac{1}{2}\log\dfrac{p_j(x)}{1-p_j(x)} \in R$

Donde:

$p_j(x) = p(G = 1/x)$

……

Algoritmo2.3: Adaboosting Real para 2 clases.

1.- Inicializamos las observación con pesos $v_i = 1/N$,i=1,…,N

2.- Repetir j=1 hasta T

a) Ajuste del clasificador para obtener una estimación de probabilidad a la clase $p_j(x) = p(G = 1/x) \in [0,1]$, usando un conjunto de entrenamiento modificada.

b) Calcular $G_j(X) \leftarrow \frac{1}{2}\log\frac{p_j(x)}{1-p_j(x)} \in R$

c) Calcular $v_i \leftarrow v_i.\exp\left[-G_i.G_j(x_i)\right]$ i=1,..,N y normalizado $\sum_{i=1}^{N} v_i = 1$

3.- Salida del clasificador combinado.

$$G_{com}(x) = sign\left(\sum_{j=1}^{T} c_j G_j(x)\right)$$

.....

El algoritmo 2.4 que a continuación presentamos fue propuesto por Bauer y Kohavi y es una generalización para varias clases [3].

……

Algoritmo 2.4: adaboosting multiclase.

Entrada: Una muestra de entrenamiento E de tamaño N, un clasificador G(x) , T numero de adaboosting(Numero de clasificadores base).

1.- Inicializamos las observación con pesos $v_1(x_i) = 1/N$,i=1,…,N

2.-Desde j=1,.., T **{**

3.- Entrenar la red con la finalidad de construir un clasificador $G_j(x)$

4.- Calculo del error ponderado $err_j = \sum_{i=1}^{N} v_i(x_i).I(G_j(x_i) \neq G_i)$

5.- Si $err_j \succ 1/2 \ \vee \ err_j = 0$ entonces, $j = j-1$

6.- Calculo de los coeficientes $c_j = err_j/(1-err_j)$

7.- Actualización de los pesos para cada observación, x_i

$$v_{i+1}(x_i) = \begin{cases} v_i(x_i)/2(err_j) & si \quad G_j(x_i) = G_i \\ v_i(x_i)/2(1-err_j) & si \quad G_j(x_i) \neq G_i \end{cases}$$

8.-Normalizar los pesos. **}**

9.- $G_{com}(x) = \arg\max_{G \in C} \sum_{j=1}^{T} \log\left(\frac{1}{c_j}\right) I(G_j(x) = G) \qquad C = \{1,2,..,K\}$

10.- Salida de la combinación de clasificadores adaboosting.

……

Estos algoritmos de adaboosting inicialmente asignan pesos iguales a las observaciones. Cada vez que se genere un clasificador, se cambian los pesos usados para el siguiente clasificador. La idea es forzar al nuevo clasificador a minimizar el error. Para esto se asigna más peso a las observaciones mal clasificadas y menos a las correctamente clasificadas. El clasificador final se forma usando votación ponderada.

2.1.2.2 Adaboosting como un Modelo Aditivo.

La predicción final de la combinación de clasificadores Adaboosting [9] puede expresarse como una expansión aditiva de T funciones base, $h_j(x)$, con sus respectivos coeficientes de expansión b_j .esto es:

$$f(x) = \sum_{j=1}^{T} b_j h_j(x) \qquad (2.37)$$

El lenguaje de adaboosting $h_j(x)$ representa al clasificador base caracterizados por un conjunto de parámetros (pesos w) y posee un argumento

multivariado en x y $f(x)$ es la predicción final de la combinación de clasificadores adaboosting.

En el caso de las redes neuronales, los clasificadores base tienen la forma particular: $h_j(x) = s(W^T x)$, donde: $s(.)$ es la función de transferencia.

Trevor Hastie, Robert Tibshirani, Jerone Friedman 2000 consideran Adaboosting desde la perspectiva estadística como un método para ajustar una expansión aditiva $f(x) = \sum_{j=1}^{T} b_j h_j(x)$ en un conjunto de "funciones base" mediante "forward stagewise". Las funciones base, en este caso, son los clasificadores base $h_j(x) \in \{-1,1\}$, por conveniencia expresaremos $G_j(x) = s(W^T x)$ en lugar $h_j(x)$ para referirnos al clasificadores base.

Típicamente estos modelos son ajustados minimizando la función pérdida, sobre el conjunto de entrenamiento mediante:

$$\{b_j, W_j\}_{j=1}^{T} = \min_{\{b_j, W_j\}_{j=1}^{T}} \sum_{i=1}^{N} L\left(y_i, \sum_{j=1}^{T} b_j s(W^T x_i)\right) \qquad (2.38)$$

Esta minimización requiere de algoritmos de optimización con un costo de tiempo muy alto. Sin embargo es posible resolver el problema de ajuste considerando sub-problemas de funciones base, de la forma:

$$\{b, W\} = \min_{b,W} \sum_{i=1}^{N} L\left(y_i, b.s(W^T x_i)\right) \qquad (2.39)$$

2.1.2.3 Forward stagewise.

Es un método computacional que permite aproximar la solución de (2.38), mediante una secuencia de ajuste de modelos base $f_1(x),..., f_{j-1}(x)$. [9]

Forward stagewise es un proceso iterativo y puede ser expresado de la siguiente forma:

$$f_j(x) = f_{j-1}(x) + b_j s(W^T x) \quad (2.40)$$

En el cual típicamente, nosotros minimizamos una función de pérdida para una muestra de entrenamiento.

……

Algoritmo2.5: Forward stagewise

1.- Inicializar $f_0(x) = 0$

2.-Repetir j =1 hasta T

a) Calcular

$$(b_j, W_j) = \arg\min_{b,W} \sum_{i=1}^{N} L\left(y_i, f_{j-1}(x_i) + b.s(W^T x_i\right)$$

b) $f_j(x) = f_{j-1}(x) + b_j s(W^T x) = f_{j-1}(x) + b_j G_j(x)$

……..

El algoritmo 2.5 en cada iteración j, obtiene una solución para las funciones base $s(W_j^T X)$ y su correspondiente coeficiente de expansión b_j. En base a estos resultados y $f_{j-1}(x)$ se obtiene $f_j(x)$.

La forma de la función de pérdida, $L\left(y_i, f_{j-1}(x_i) + b.s(W^T x_i\right)$ varía de acuerdo a la aplicación que se tenga. En el caso de regresión, se usa la función de pérdida cuadrática y para clasificación la función de pérdida exponencial.

2.1.2.4 Adaboosting y función de pérdida exponencial.

Actualmente existen trabajos [9] que muestran que adaboosting es equivalente al algoritmo forward stagewise de modelos aditivos usando función de pérdida exponencial , $L(G, f(x)) = \exp(-G.f(x))$.

En el algoritmo forward stagewise obtenemos en cada iteración j, $f_j(X)$ que minimice el error de entrenamiento.

$$f_j(x) = \arg\min_{f_j} \sum_{i=1}^{N} L(G_i, f_j(x_i)) , \quad \text{j=1,…,T} \tag{2.41}$$

como cada una de las $f_j(x)$, $j = 1,...,T$, está en función de su clasificador base y coeficiente, esto es, $f_j(x) = f_{j-1}(x) + b_j.G_j(x)$, entonces el proceso de minimización se realiza en base al clasificador y el coeficiente .

Usando la función de pérdida exponencial y reemplazando $f_j(x)$, la ecuación (2.41) se puede expresar de la siguiente forma:

$$(b_j, G_j(x)) = \arg\min_{b_j, G_j(x)} \sum_{i=1}^{N} \exp\left(- G_i\left[f_{j-1}(x_i) + b_j.G_j(x_i)\right]\right) \tag{2.42}$$

$$(b_j, G_j(x)) = \arg\min_{b_j, G_j(x)} \sum_{i=1}^{N} \exp\left(G_i.f_{j-1}(x_i)\right).\exp(-b_j.G_i.G_j(x_i)) \tag{2.43}$$

Considerando $v_i^j = \exp(-G_i.(f_{j-1}(x_i))$, este valor depende de $f_{j-1}(x)$.

La solución de (2.43) puede ser obtenida en dos etapas. Primero para algún valor de b >0, el clasificador se obtiene mediante:

$$G_j(x) = \arg\min_{G_j(x)} \sum_{G_i = G_j(x_i)} v_i^j.e^{-b} + \sum_{G_i \neq G_j(x_i)} v_i^j.e^{b} \tag{2.44}$$

$$G_j(x) = \arg\min_{G_j(x)} (e^{b} - e^{-b}) \sum_{i=1}^{N} v_i^j I(G_i \neq G_j(x_i)) + e^{-b} \sum_{i=1}^{N} v_i^j \ , \tag{2.45}$$

dividiendo entre $\sum_{i=1}^{N} v_i^j$, en la ecuación anterior se tiene:

$$G_j(x) = \arg\min_{G_j(x)} (e^{b} - e^{-b}) \frac{\sum_{i=1}^{N} v_i^j I(G_i \neq G_j(x_i))}{\sum_{i=1}^{N} v_i^j} + e^{-b} \tag{2.46}$$

$$G_j(x) = \arg\min_{G_j(x)} (e^{b} - e^{-b}) err_j + e^{-b} \ , \tag{2.47}$$

donde $err_j = \frac{\sum_{i=1}^{N} v_i^j . I(G_i \neq G_j(x_i))}{\sum_{i=1}^{N} v_i^j}$, es el error de entrenamiento ponderado.

En la segunda etapa, nuestro objetivo ahora es estimar los coeficientes b_j, para ello minimizamos

$$\frac{\partial G_j(x)}{\partial b_j} = (e^{b_j} + e^{-b_j}) err_j - e^{-b_j} = 0 , \quad (2.48)$$

de donde tenemos:

$$b_j = \frac{1}{2} \log \frac{1 - err_j}{err_j} \quad (2.49)$$

Finalmente la aproximación es actualizada mediante:

$$f_j(x) = f_{j-1}(x) + b_j G_j(x) , \quad (2.50)$$

y los pesos para la siguiente iteración esta dado por:

$$v_i^{j+1} = v_i^j . e^{-b_j . G_i G_j(x_i)} \quad (2.51)$$

$$v_i^{j+1} = v_i^j . e^{c_j . I(G_i \neq G_j(x_i))} . e^{-b_j} \text{ ,donde } c_j = 2b_j , \quad (2.52)$$

puesto que:

$$-G_i G_j(x_i) = 2.I(G_i \neq G_j(x_i)) - 1 \quad (2.53)$$

$c_j = 2b_j$ es el coeficiente definida en el algoritmo adaboosting y $\exp(b_j)$ se cancela cuando se normaliza los valores de los pesos v_i. Con esto se demuestra que el algoritmo forward stagewise de modelos aditivos basados en pérdida exponencial, es equivalente al adaboosting (Algoritmo 2.2).

Por otra parte el algoritmo 2.3, puede ser visto como un proceso de estimación mediante forward stagewise usando el criterio exponencial $E(\exp(-G.f(x))$.

$$f^*(x) = \arg\min_{f(x)} E(\exp(-G.f(x)) \quad , \; Ge\{-1,1\} \tag{2.54}$$

$$E(\exp(-G.f(x)) = p(G=1/x).\exp(-f(x) + p(G=-1/x).\exp(f(x), \tag{2.55}$$

derivando, se tiene:

$$\frac{\partial}{\partial}\frac{E(\exp(-G.f(x))}{f(x)} = -p(G=1/x).\exp(-f(x) + p(G=-1/x).\exp(f(x) = 0 \tag{2.56}$$

$$f^*(x) = \frac{1}{2}\log\frac{p(G=1/x)}{p(G=-1/x)} \tag{2.57}$$

2.1.1.1 Adaboosting y función de pérdida cuadrática.

El método adaboosting para problemas de regresión se obtiene del algoritmo forward stagewise, usando la función pérdida cuadrática [9].

$$L(y_i, f(x_i)) = (y_i - f_{j-1}(x_i) - b.s(W^T x_i))^2 \text{, i =1,…,N} \tag{2.58}$$

Donde $r_{ij} = y_i - f_{j-1}(x_i)$ es el residual.

……

Algoritmo 2.6 : adaboosting para regresión

1.- Inicializar $f_0(x) = \bar{y}$

2.-Repetir j =1 hasta T

a) Calcular $(b_j, W_j) = \arg\min_{b,W} \sum_{i=1}^{N} (r_{ij} - b.s(W^T x_i))^2$

b) $f_j(x) = f_{j-1}(x) + b_j s(W^T x) = f_{j-1}(x) + b_j G_j(x)$

……

2.2 Boosting para redes neuronales.

El modelo general de una red neuronal con vector de parámetros $\boldsymbol{q}=(\boldsymbol{b},w)$, es una función no lineal dado por:

$$f(X,q)=g^{s}\left[b_{0k}+b_{k}^{T}s\left(w_{0m}+w_{m}^{T}X\right)\right], \tag{2.59}$$

donde g^{s} , s son funciones de transferencia y s es el número de capas ocultas que posee la red.

Los valores de los parámetros son obtenidos por el método de backpropagation, minimizando la función de error:

$$q=\arg\min_{q} R(q);R(q)=\sum_{i=1}^{N}L(y_{ik},f_{k}(x_{i}))=\begin{cases}\sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik},f_{k}(x_{i}))^{2} & \text{Regresión}\\ \sum_{i=1}^{N}\sum_{k=1}^{K}G_{ik}\log f_{k}(x_{i}) & \text{Clasificación}\end{cases} \tag{2.60}$$

$f_{k}(x_{i})$ Representa un valor de $f(X,q)$ para una observación particular x y $L(.)$ la función de pérdida.

El modelo boosting para redes neuronales se puede expresar en forma similar a (2.37), mediante una suma de T redes base [15].

$$f_{boos}=\sum_{j=1}^{T}f(X,q_{j}) \tag{2.61}$$

Los parámetros son obtenidos minimizando la función perdida, sobre el conjunto de entrenamiento mediante:

$$\{q_{j}\}_{j=1}^{T}=\min_{\{q_{j}\}_{j=1}^{T}}\sum_{i=1}^{N}L\left(y_{i},\sum_{j=1}^{T}f(X,q_{j})\right) \tag{2.62}$$

En cada iteración de este algoritmo se estima los parámetros del j-ésimo modelo mediante:

$$q_j = \arg\min_{q_j} \sum_{i=1}^{N} L\left(y_i, f_{j-1}(x_i) + f(x, q_j)\right) \qquad (2.63)$$

En la sección anterior para problemas de clasificación de dos clases y función de perdida exponencial, nosotros mostramos que la solución de (2.63), es la red que minimiza el error ponderado $err_j = \sum_{i=1}^{N} v_i^j . I(G_i \neq G_j(x_i)) / \sum_{i=1}^{N} v_i^j$, con pesos $v_i^{j+1} = v_i^j . \exp(-b_j . G_i G_j(x_i))$.

2.3 Sesgo y varianza de una combinación de clasificadores.

Una de las razones de la combinación de clasificadores es que estos son más precisos que los clasificadores base reduciendo la tasa de error, sesgo y varianza.

Breiman (1996) mostró experimentalmente que bagging reduce la varianza, esto se debe a que bagging puede ser vista como un método de clasificación usando una estimación de tendencia central, mientras que Freund y Shapire (1996) sostienen que boosting reduce el error en el termino sesgo y varianza.

En la sección 2.3 se demostró que la combinación de clasificadores $f_{com}(x) = \sum_{j=1}^{T} c_j f_j(x)$, reduce el error y por ende la varianza. En el caso particular de que $c_j = \frac{1}{T} \ \forall j$, se tiene la combinación de clasificadores bagging y cuando c_j toma diferentes valores se tiene la combinación de clasificadores

boosting, por tanto ambos métodos de clasificación bagging y boosting reducen la varianza.

El error de la combinación se puede descomponer en forma similar a la ecuación 1.126 , esto es:

$$E\left(f_{com}(x)-Y\right)^2 = E\left(Y-f(x)\right)^2 + \left(E\left(f_{com}(x)\right)-f(x)\right)^2 + E\left(f_{com}(x)-E\left(f_{com}(x)\right)\right)^2 \quad (2.64)$$

El primer término de la ecuación anterior corresponde al error irreducible, el cual es independiente de los parámetros del modelo y por tanto podemos omitir este término en (2.64)

$$E\left(f_{com}(x)-Y\right)^2 = \left(E\left(f_{com}(x)\right)-f(x)\right)^2 + E\left(f_{com}(x)-E\left(f_{com}(x)\right)\right)^2 \quad (2.65)$$

$$E\left(f_{com}(x)-Y\right)^2 = \left[ses\left(f_{com}(x)\right)\right]^2 + \mathrm{var}\left(f_{com}(x)\right) \quad (2.66)$$

Considerando que los modelos entrenados por la red para resolver un problema tienen la misma importancia, el modelo de la combinación está dado por:

$$f_{com}(x) = \frac{1}{T}\sum_{j=1}^{T} f_j(x) \quad (2.67)$$

y el sesgo y varianza de la combinación de clasificadores mediante:

$$\mathrm{var}\left(f_{com}(x)\right) = \mathrm{var}\left(\frac{1}{T}\sum_{j=1}^{T} f_j(x)\right) = E\left\{\left(\frac{1}{T}\sum_{j=1}^{T} f_j(x) - E\left\{\frac{1}{T}\sum_{j=1}^{T} f_j(x)\right\}\right)^2\right\} \quad (2.68)$$

$$= E\left\{\left(\frac{1}{T}\sum_{j=1}^{T}\left\{f_j(x)-E\left(f_j(x)\right)\right\}\right)\left(\frac{1}{T}\sum_{i=1}^{T}\left\{f_i(x)-E\left(f_i(x)\right)\right\}\right)\right\}$$

$$= E\left\{\left(\frac{1}{T^2}\sum_{j=1}^{T}\sum_{i=1}^{T}\left\{f_j(x)-E\left(f_j(x)\right)\right\}\left\{f_i(x)-E\left(f_i(x)\right)\right\}\right)\right\}$$

$$\mathrm{var}\left(f_{com}(x)\right) = E\left\{\left(\frac{1}{T^2}\sum_{j=1}^{T}\sum_{i=1,i\neq j}^{T}\left\{f_j(x)-E\left(f_j(x)\right)\right\}\left\{f_i(x)-E\left(f_i(x)\right)\right\}\right)\right\}$$

$$+ E\left\{\left(\frac{1}{T^2}\sum_{i=j}^{T}\{f_i(x) - E(f_i(x))\}\right)^2\right\} \qquad (2.69)$$

En forma similar el sesgo de la combinación es:

$$ses(f_{com}(x)) = ses\left(\frac{1}{T}\sum_{j=1}^{T} f_j(x)\right) = \left(E\left\{\frac{1}{T}\sum_{j=1}^{T} f_j(x)\right\} - f(x)\right)^2 = \left(\frac{1}{T}\sum_{j=1}^{T} E(f_j(x) - f(x))\right)^2 \qquad (2.70)$$

Considerando todo los conjuntos de entrenamiento E_j, $j = 1,..,T$, definimos el sesgo, varianza y covarianza promedio de los miembros de la combinación de clasificadores mediante:

$$sesgo_{prom}(x) = \frac{1}{T}\sum_{j=1}^{T}\left(E(f_j(x)) - f(x)\right) = \frac{1}{T}\sum_{j=1}^{T} ses(f_j(x)),\ f(x) = E(Y/x) \qquad (2.71)$$

$$\mathrm{var}_{prom}(x) = \frac{1}{T}\sum_{j=1}^{T} E\left\{\left(f_j(x) - E(f_j(x))\right)^2\right\} = \frac{1}{T}\sum_{j=1}^{T}\mathrm{var}(f_j(x)) \qquad (2.72)$$

$$\mathrm{cov}_{prom}(x) = \frac{1}{T(T-1)}\sum_{j=1}^{T}\sum_{i=1}^{T}\mathrm{cov}(f_i(x), f_j(x)) \qquad (2.73)$$

Reemplazando (2.71), (2.72) y (2.73) en (2.69) y (2.70), la varianza y sesgo de la combinación quedan representados en función del sesgo, varianza y covarianza promedio:

$$\mathrm{var}(f_{com}(x)) = \left(1 - \frac{1}{T}\right)\mathrm{cov}_{prom}(x) + \frac{1}{T}\mathrm{var}_{prom}(x) \qquad (2.74)$$

$$ses(f_{com}(x)) = sesgo_{prom}(x) \qquad (2.75)$$

Finalmente reemplazando (2.74) y (2.75) en (2.65) , la descomposición para el error de la combinación está dada por:

$$E(f_{com}(x) - Y)^2 = \left(1 - \frac{1}{T}\right)\mathrm{cov}_{prom}(x) + \frac{1}{T}\mathrm{var}_{prom}(x) + (sesgo_{prom}(x))^2 \qquad (2.76)$$

De la ecuación anterior aparentemente el sesgo de la combinación es idéntico al sesgo de cada uno de los miembros de la combinación puesto que es igual

al sesgo representativo (promedio) y esto no se reduce. Por consiguiente, estimadores que deberían ser usados en la combinación deben tener sesgo pequeño y la covarianza también debería ser pequeña desde que este término de la combinación no se reduce con el incremento de los componentes de la combinación. Lo nuevo de este modelo es que la varianza de la combinación decrece como en $1/T$. Si nosotros tenemos estimadores con sesgo pequeño y baja covarianza entre los miembros de la combinación, la esperanza del error de la combinación disminuye significativamente con respecto a la esperanza del error de los miembros de la combinación.

En algunos casos la combinación puede usarse para reducir el sesgo y varianza. El sesgo es reducido por el diseño de la arquitectura de los miembros de la combinación, redes base, y la varianza es reducida por usar una medida de tendencia central para formar la combinación [15].

Podemos enfocar el problema de clasificación como una regresión por aproximación de probabilidades posteriores, sección 1.10.4, esto permite que la teoría que hemos desarrollado se pueda aplicar para clasificación; por ejemplo, en problemas de clasificación de dos clases, Y podría representar la probabilidad de una clase, $1-Y$ probabilidad de la otra clase y $f_j(x)$ es el

j-ésimo estimador de Y.

Desafortunadamente en la práctica no es fácil de calcular el sesgo y varianza de la combinación. Existen algunas alternativas de la descomposición del sesgo y varianza para combinación de clasificadores, uno de ellos propuesto por Breiman la cual presentamos a continuación.

2.3.1 **Caso de la regresión.**

Dado un conjunto de entrenamiento E = {(x_i, y_i) / $x_i \varepsilon \Re^p$, i=1,…N}, donde y_i representan salidas numéricas y x_i es un vector de entradas, la red neuronal es aplicada al conjunto de entrenamiento para construir un conjunto de predictores $f_j(X)$ de futuras observaciones de y.

Asumiendo que el conjunto de entrenamiento consiste de muestras independientes idénticamente distribuidas con distribución de y, X; definimos el error de predicción para el caso de regresión.

$$Err(f(X) = E\left\{(y - f(X))^2\right\} \qquad (2.77)$$

Nosotros podemos descomponer siempre y mediante:

$$y(X) = f^*(X) + e \quad \text{donde } E(y/X) = f^*(x),\ E(e) = 0,\ \text{var}(e) = s_e^2 \qquad (5.78)$$

Dado la predicción de la combinación

$$f_{com}(X) = E_E\left[f_j(X)\right] = \frac{1}{T}\sum_{j=1}^{T} f_j(X), \qquad (2.79)$$

definimos el sesgo y varianza como:

$$Sesgo(f) = E_X\left(f(X) - f_{com}(X)\right)^2 \qquad (2.80)$$

$$Var(f) = E_{E,X}\left(f_j(X) - f_{com}(X)\right)^2 \qquad (2.81)$$

la descomposición del error de predicción es dado por:

$$Err(f(X) = E\left(e^2\right) + Sesgo(f) + Var(f) \qquad (2.82)$$

2.3.2 **Caso de la clasificación.**

Utilizando redes neuronales construimos un conjunto de clasificadores $G_j(X)$ en base a T conjuntos de entrenamiento E_1, E_2,.., E_T generados a partir de E, cada uno de ellos de tamaño N. La combinación de clasificadores esta dado por $G_{com}(x_0) = \arg\max_{g \in C} p(G_j(x_0) = G_g)$ y el clasificador de Bayes mediante:

$$G_{Bayes}(x_0) = k \quad si \quad p(G = k / x_0) = \max_{g \in C} p(G = g / x_0) \tag{2.83}$$

Breiman define una partición del conjunto de entrenamiento, E, en dos conjuntos. El conjunto U, ‘’insesgado’’, consiste de todo los xeX para los cuales $G_{com}(x) = G_{bayes}(x)$ y el conjunto B, ‘’sesgado’’, dado por el complemento de U, esto es, B = E-U.

Dado estos conjuntos Breiman (1998) [6] define el sesgo y varianza basado en la idea que observaciones del conjunto B (sesgado) participen en el sesgo y las observaciones de U (insesgado) participen en la varianza.

$$sesgo(x) = p_{X,Y}\left(G_{bayes}(x) = G \quad x \in B\right) - E_E\left[p_{X,Y}\left(G_j(x) = G \quad x \in B\right)\right] \tag{2.84}$$

$$\text{var}ianza(x) = p_{X,Y}\left(G_{bayes}(x) = G \quad x \in U\right) - E_E\left[p_{X,Y}\left(G_j(x) = G \quad x \in U\right)\right], \tag{2.85}$$

donde $E_E(.)$ es tomada con respecto a todo los conjuntos de entrenamiento

Otra definición muy importante de sesgo y varianza de un clasificador fue propuesto por Kong y Dietterich(1996) [3].

$$sesgo = PE(G_{com}) - PE(G_{bayes}) \tag{2.86}$$

$$\text{var}ianza = E\left[PE(G_j)\right] - PE(G_{com}) \ , \tag{2.87}$$

donde $PE(G) = P_{X,Y}\left(G(x) = G\right)$ es el error esperado de un clasificador.

CAPÍTULO III: APLICACIÓN.

3.1 Introducción.

En este capítulo se presentan resultados al aplicar el método de redes neuronales y los métodos de combinación de clasificadores bagging y boosting basados en esta teoría para resolver problemas de clasificación. Los métodos se aplicaron a dos bases de datos: sonar e iris, que fueron tomadas del "Machine Learning Database Repository" [www.ics.uci.edu~mlearn] disponible en la Universidad de California en Irvine y se resumen en la Tabla 1. Estos conjuntos de datos se han usado en otros trabajos relacionados con clasificación, esto permite comprobar la bondad de los resultados del presente trabajo.

Tabla 1. Información a cerca del conjunto de datos usados

Data	**Número de observaciones**	**Número de clases**	**Número de variables continuas**
Iris	**150**	**3**	**4**
Sonar	**208**	**2**	**60**

Este capítulo está organizado en 3 secciones, en la primera sección se presentan y discuten los resultados de la red neuronal. En la segunda sección se presentan y discuten los métodos "bagging" y boosting. Finalmente en la tercera sección se presentan resultados para comparar los dos métodos de combinación de clasificadores aplicados a los dos conjuntos de datos.

3.2 Análisis y presentación de resultados.

En la aplicación del presente trabajo se utilizó el programa de data mining TANAGRA versión 1.4.12 [http://eric.univ-lyon2.fr/~rico/tanagra/] ; con los propósitos de: entrenar la red neuronal, determinar el error de entrenamiento ,

analizar la matriz de confusión y las tasas de error de mala clasificación . A partir de ellas se efectúan los análisis y comentarios respectivos para cada caso y obtener finalmente las conclusiones válidas y sustentadas para la investigación.

3.2.1 Base de datos Iris.

La base de datos iris corresponde a características de flores, contiene 150 casos, 4 variables cuantitativas: Largo del sépalo, Ancho del sépalo, Largo del pétalo, Ancho del pétalo y una variable cualitativa (tipo de rosa) con 3 categorías: Iris setosa, Iris versicolor e Iris virginica.

Antes de entrenar la red neuronal debemos recordar lo siguiente. A medida que el número de: capas ocultas, neuronas y interacciones crece se tiene una mayor posibilidad de sobreajuste de la red a los datos de entrenamiento y generando como consecuencia de ello una mayor varianza. Otro factor que se debe tener en cuenta es el factor de aprendizaje, valores cercanos a uno generan una estimación deficiente de los pesos de la red, debido a que los pesos son estimados con un error local y no con un error global.

En el proceso de aprendizaje de la red se a probado con 3 capas, en la capa oculta se evaluó con distinto número de neuronas, resultando adecuado (menor error) con 13 neuronas y un factor de aprendizaje de 0.15.

Finalmente la red neuronal es entrenada utilizando las siguientes características: 3 capas (entrada, oculta y de salida). La capa oculta con 13 neuronas, el factor de aprendizaje es 0,15 y la regla de parada en el proceso de aprendizaje es de 100 interacciones o una tasa de error de mala clasificación de 0.01. A continuación se muestra el error de entrenamiento de algunas interacciones.

Tabla 2. Evaluación del error

Interacción	Entrenamiento	Validación	Error
1	0.6833	0.6000	40.7093
6	0.6833	0.6000	39.4005
11	0.3417	0.3333	26.4713
16	0.1500	0.2000	20.6984
21	0.1250	0.1667	18.7602
26	0.1250	0.1667	16.7890
31	0.1167	0.2333	14.8170
36	0.1333	0.2333	13.4449
41	0.1167	0.2000	12.4107
46	0.1250	0.1667	11.4558
51	0.1083	0.1667	10.4703
56	0.0917	0.1333	9.4364
61	0.0750	0.0667	8.4116
66	0.0583	0.0667	7.4856
71	0.0333	0.0667	6.7139
76	0.0333	0.0667	6.1000
81	0.0333	0.0667	5.6189
86	0.0333	0.0667	5.2411
91	0.0333	0.0667	4.9407
96	0.0333	0.0667	4.6984

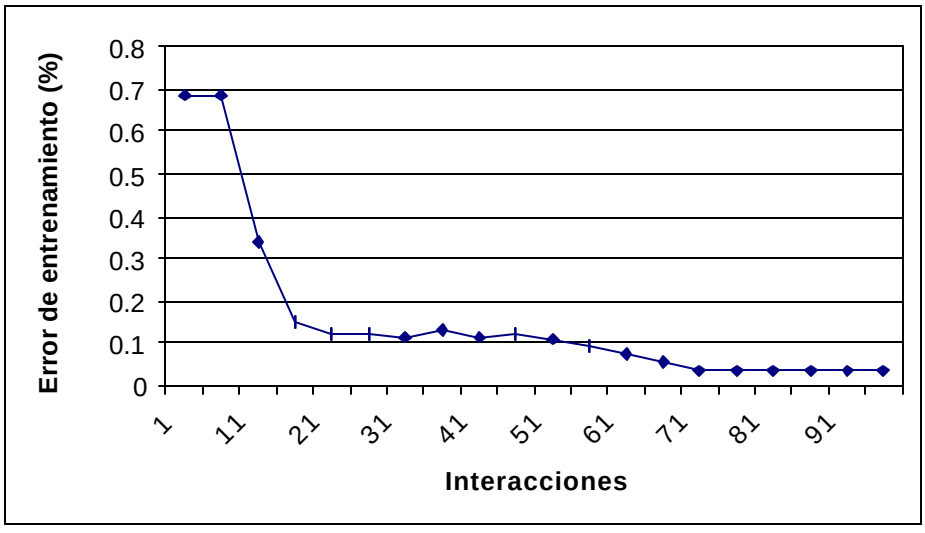


Figura 3.1 Comportamiento del error de entrenamiento.

En la Fig.3.1 se observa el comportamiento del error de entrenamiento de la red, de ésta se puede concluir que el error decrece a medida que la red aprende a reconocer el comportamiento de los datos. A menor error de entrenamiento mayor ajuste de los datos a la red.

Para medir la bondad de la red neuronal es necesario calcular la matriz de confusión.

Tabla 3. Matriz de confusión.

Tasa de error.	0.0400

Grupo	Precisión
Setosa	0.0000
versicolor	0.0600
Virginica	0.0600

	Setosa	versicolor	Virginica	Suma
Setosa	50	0	0	50
versicolor	0	47	3	50
Virginica	0	3	47	50
Sum	50	50	50	150

La Tabla 3 muestra la matriz de confusión del clasificador red neuronal para la base de datos iris, de ésta se puede observar que el modelo es adecuado en un 96% de los casos.

Una de las formas alternativas de la reducción de la tasa de error obtenida usando redes neuronales es el uso de combinación de clasificadores tal es el caso de bagging y boosting, para este fin se evaluó distintos números de clasificadores base, estabilizándose la tasa de error para el método boosting desde 30 clasificadores, por lo cual se decidió utilizar 30 clasificadores base.

Tabla 4. Tasa de error de entrenamiento

Numero clasificadores	Tasa de error de Bagging	Tasa de error de Boosting
1	0.0467	0.0733
2	0.26	0.06

3	0.3667	0.0533
4	0.1933	0.0533
5	0.16	0.0533
6	0.1667	0.0533
7	0.08	0.0533
8	0.4133	0.0467
9	0.42	0.0467
10	0.24	0.0467
11	0.0733	0.0467
12	0.1267	0.0467
13	0.38	0.04
14	0.04	0.04
15	0.0933	0.04
16	0.12	0.04
17	0.0867	0.04
18	0.4333	0.04
19	0.4333	0.04
20	0.24	0.04
21	0.08	0.0333
22	0.3333	0.0333
23	0.06	0.0333
24	0.1667	0.0333
25	0.1667	0.0333
26	0.38	0.0267
27	0.1	0.0267
28	0.3933	0.0267
29	0.0533	0.0267
30	0.5267	0.02

En la figura siguiente, se muestra las tasas de error de los 30 clasificadores individuales.

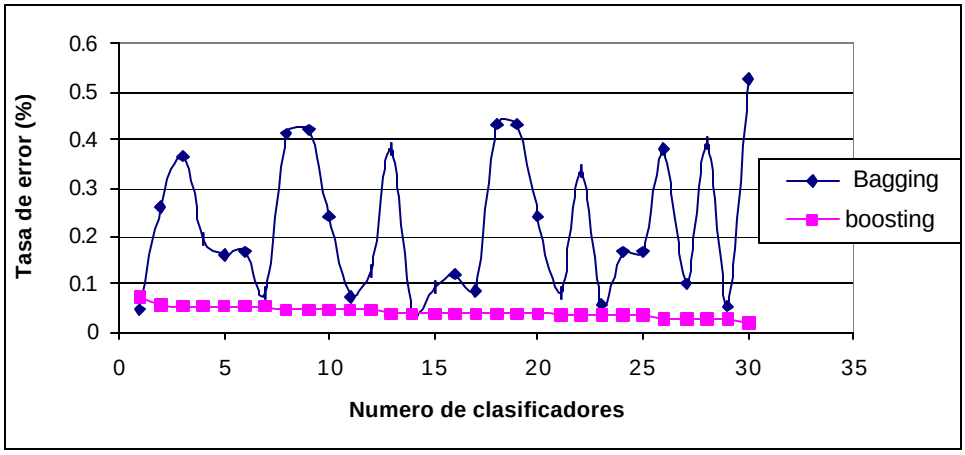


Figura 3.2. Comparación de la tasa de error de los clasificadores bases utilizados para formar bagging y boosting

En la figura 3.2 se observa una comparación de las tasas de error de los clasificadores usados para los métodos de combinación bagging y boosting. Para el caso del método bagging las 30 muestras de entrenamiento se obtienen por validación cruzada y en cada una de estas muestras se entrena la red en forma paralela, por tal razón el comportamiento de la tasa de error es variable (aleatorio), en cambio, para el método boosting las muestras se entrenan en forma secuencial, esto es, primero se entrena una muestra de entrenamiento, las observaciones clasificadas incorrectamente se ponderan con pesos pequeños y a las observaciones correctamente clasificadas se les asigna pesos altos. Con esta muestra modificada se entrena nuevamente la red obteniendo un segundo clasificador y así sucesivamente se obtienen todo los clasificadores.

La forma de entrenar las muestras con el método boosting permite que las tasas de error de los clasificadores tiendan a decrecer en comparación al clasificador anterior.

La matriz de confusión de los métodos de bagging y boosting se muestra a continuación.

Tabla 5. Matriz de confusión.

Bagging				
Tasa de error =0.0333				
	Setosa	**versicolor**	**Virginica**	**Suma**
Setosa	50	0	0	50
versicolor	0	47	3	50
Virginica	0	2	48	50
Sum	50	49	51	150

Boosting			
Tasa de error =0.0267			
Setosa	**versicolor**	**Virginica**	**Suma**
50	0	0	50
0	48	2	50
0	2	48	50
50	50	50	150

Número de Clasificadores 30

En el cuadro anterior se muestra la matriz de confusión de los métodos de combinación de clasificadores bagging y boosting. Se observa claramente que la bondad de los métodos bagging (96.7%) y boosting (97.33%) son relativamente mejores que el clasificador base (red neuronal) de 96%.

El clasificador bagging se obtiene por votación mayoritaria de los clasificadores base, pero dando la misma importancia, a cada clasificador en cambio el clasificador boosting da mayor importancia a los clasificadores que presentan mayor eficiencia (menor tasa de error de mala clasificación). Es decir el bagging funciona como el promedio aritmético de los clasificadores y el boosting como un promedio ponderado, donde las ponderaciones son la importancia de los clasificadores.

3.2.2 Base de datos Sonar

Este conjunto de datos consta de 208 muestras de señales de sonido con 60 variables predictoras y dos clases que corresponden a dos tipos de señales. Un grupo de 111 señales emitidas de cilindros metálicos y 97 muestras emitidas en piedras cilíndricas. Las características de las muestras se hicieron bajo

diversas condiciones en los ángulos y en diversas condiciones. Cada valor de la muestra representa la energía en una banda de frecuencia particular integrada sobre cierto periodo de tiempo.

Para estimar la tasa de error de mala clasificación se entrena la red neuronal con 3 capas y en la capa oculta se utilizó 15 neuronas, el factor de aprendizaje es 0,01 y la regla de parada en el proceso de aprendizaje es de 500 interacciones o una tasa de error de mala clasificación de 0.01, tal como se ilustra en el cuadro siguiente.

Tabla 6. Evaluación del error.

Interacción	Entrenamiento	Validación	Error
1	0.4819	0.4048	41.4525
26	0.4819	0.4048	40.7831
51	0.2108	0.1905	29.7081
76	0.1386	0.2143	20.5300
101	0.1084	0.1905	16.9798
126	0.1084	0.2143	14.8600
151	0.0723	0.2381	13.3334
176	0.0542	0.2381	12.2074
201	0.0542	0.2381	11.3008
226	0.0482	0.2381	10.4860
251	0.0482	0.2381	9.7041
276	0.0422	0.2381	9.0683
301	0.0422	0.2381	8.5752
326	0.0361	0.2381	8.1855
351	0.0361	0.2381	7.8729
376	0.0361	0.2381	7.6187
401	0.0361	0.2381	7.4090
426	0.0361	0.2381	7.2333
451	0.0361	0.2381	7.0835
476	0.0361	0.2381	6.9529

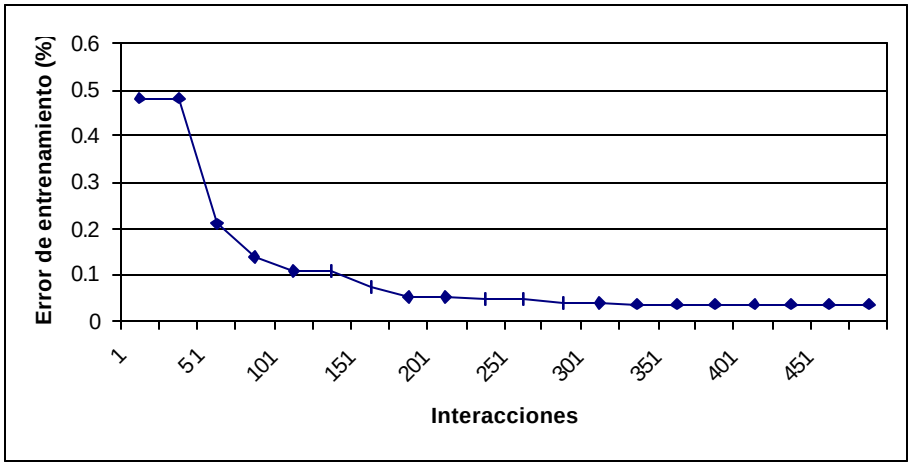


Figura 3.3 Evolución del error de entrenamiento

El error de entrenamiento decrece hasta la interacción 200 (repetición de entrenamiento de la red), después de esta interacción tiende a estabilizarse, por tanto, con 200 interacciones ya se puede estimar el modelo de la red.

El error de entrenamiento decrece debido a que los pesos que se da para cada observación se modifican en función al resultado que se obtuvo en el clasificador generado en la interacción anterior y esto hace que la red tienda a reconocer más la base de datos en las siguientes interacciones y como consecuencia de ello los clasificadores posteriores presenta mayor poder de clasificación.

En la tabla siguiente se muestra la tasa de clasificación.

Tabla 7. Matriz de confusion.

Tasa de error.	0.0769

Grupo	Precisión
Roca	0.0909
Mina	0.0642

	Roca	Mina	Suma
Roca	90	7	97
Mina	9	102	111
Sum	99	109	208

El clasificador obtenido utilizando el método de redes neuronales con la base de datos sonar presenta una bondad de 92.31%.

Se aplica los métodos de combinación de clasificadores bagging y boosting con el propósito de mejorar la bondad del clasificador base (red neuronal)

Tabla 8. Tasa de error de entrenamiento.

Clasificador	Tasa de error Bagging	Tasa de error Boosting
1	0.0962	0.1731
2	0.1971	0.1442
3	0.2837	0.1394
4	0.274	0.1394
5	0.1971	0.1394
6	0.2933	0.1394
7	0.2548	0.1346
8	0.1731	0.1346
9	0.3413	0.1346
10	0.2644	0.1298
11	0.2885	0.1298
12	0.2404	0.125
13	0.2692	0.125
14	0.2692	0.1202
15	0.3462	0.1202
16	0.2404	0.1202
17	0.2404	0.1154
18	0.3365	0.1154
19	0.4087	0.1154
20	0.2933	0.1106
21	0.351	0.1106
22	0.2404	0.1106
23	0.1683	0.1058
24	0.2356	0.1058
25	0.1923	0.101

En la figura 4 se muestra el comportamiento de las tasas de error para los métodos de combinación de clasificadores bagging y boosting.

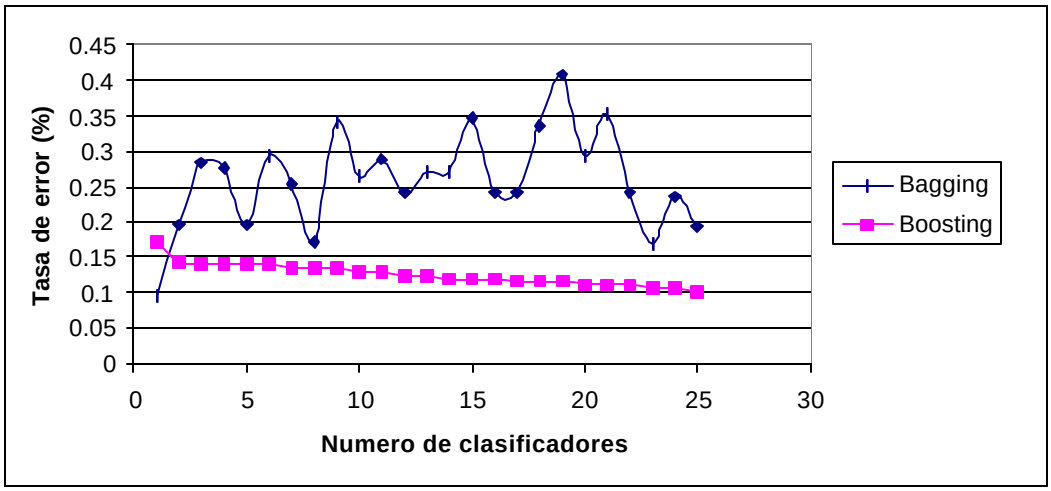


Figura 3.4 Tasa de error de mala clasificación

El comportamiento de la tasa de error de los 25 clasificadores obtenidos aplicando la red neuronal con la base de datos sonar es similar al que se obtuvo con la base de datos iris. Estos clasificadores se utilizarán para formar los clasificadores combinados bagging y boosting .

Tabla 9. Tasa de error de mala clasificación.

Boosting				Bagging		
Tasa de error =0.000				Tasa de error =0.0433		
	Roca	**Mina**	**Suma**	**Roca**	**Mina**	**Suma**
Roca	97	0	97	92	5	97
Mina	0	111	111	4	107	111
Sum	97	111	208	96	112	208
Número de Clasificadores 25						

El método de combinación de clasificadores boosting presenta una bondad del 100% para esta base de datos, en cambio el método bagging tiene una bondad

del 95.67%, estos clasificadores presentan un mayor poder de clasificación en comparación de la red neuronal.

3.3 Estudio comparativo de los métodos.

Uno de los objetivos del presenta trabajo es analizar el comportamiento de los clasificadores bagging y boosting para problemas de clasificación. En el gráfico siguiente se muestra la tasa de mala clasificación para las dos bases de datos.

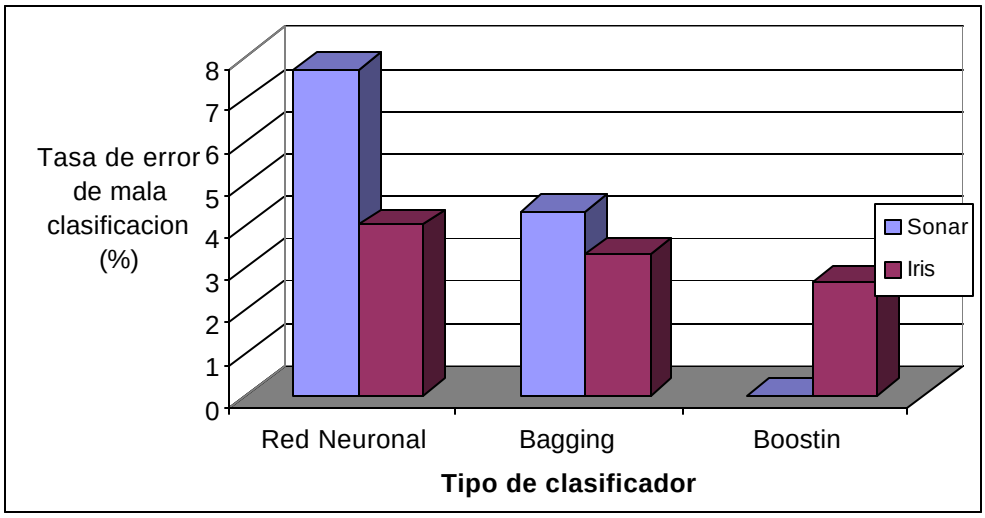


Figura 3.5. Análisis comparativo de las dos bases de estudio.

El error de mala clasificación de la red neuronal es mayor que los clasificadores combinados, en cambio el error de la clasificación del método bagging es mayor al del método boosting en las dos bases de datos utilizadas en este trabajo.

En el cuadro siguiente se muestra la comparación de la eficiencia de los clasificadores bagging y boosting con el clasificador simple (red neuronal), utilizando las bases de datos sonar e iris.

Tabla 10. Comparación de la eficiencia de los clasificadores combinados.

	Clasificador				
Data	Single	Bagging	Eficiencia	Boosting	Eficiencia
Iris	4	3.3	0.175	2.67	0.333
Sonar	7.69	4.33	0.437	0	1

Bagging es más eficiente en $|((3.3-4)/4)*100\%| = 17.5\%$ que la red neuronal en el caso de los datos iris y en 43.7% en el caso de la data sonar en cambio el método boosting es más eficiente en 33.3% de los casos en el caso de los datos iris y en 100% de los casos en sonar.

Utilizando estas bases de datos se puede concluir que el método de combinación de clasificadores boosting es más eficiente en comparación que el clasificador bagging.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. La red neuronal es un modelo estadístico no lineal, adecuado para solucionar problemas de clasificación debido a que tiene un poder de clasificación superior al 92%, como se mostró en los resultados de las dos bases de estudio.
2. Las técnicas de combinación de clasificadores: bagging y boosting en redes neuronales, reducen la tasa de mala clasificación. El nivel de reducción depende de la característica de los datos, si es homogénea menor nivel de reducción; en cambio si es heterogénea mayor nivel de reducción.
3. Entre los métodos "bagging" y "boosting" , el método boosting presenta mayor poder de clasificación, puesto que este método da mayor importancia a sus clasificadores con menor tasa de error y menor importancia a aquellos clasificadores con mayor tasa de error, en cambio, el método bagging da la misma importancia a todos sus clasificadores.

Recomendaciones

1. Se recomienda la aplicación de las técnicas de combinación de clasificadores bagging y boosting a los distintos métodos de clasificación tales como: ‘‘núcleos de kernels ‘’, funciones de gaussianas, ‘‘support vector machines ‘’ (SVM), árboles, análisis discriminante y otros.
2. La red neuronal también puede ser usada como un método de reducción del número de variables.
3. Se recomienda el uso de redes neuronales en problemas con bases de datos de tamaños grandes, en donde la distribución de los datos no es conocida.
4. Los métodos Bagging y Boosting son recomendables para solucionar problemas de clasificación donde existen categorías con muy pocos elementos.

APÉNDICES

APÉNDICE A

A1: CLASIFICADOR LINEAL.

Podemos enfocar el problema de clasificación desde el punto de vista de la regresión, para ello previamente codificamos la respuesta categórica, si G tiene K clases entonces podemos definir K indicadores de la siguiente forma:

$$Y_k = \begin{cases} 1 & si \quad G = k \\ 0 & si \quad G \neq k \end{cases}$$

de esta forma las respuestas están dadas por un vector de K componentes, $Y = (Y_1,...,Y_K)$. Utilizando la estimación del modelo de regresión mediante redes simples tenemos:

$f_k(x) = g(w_0 + w_k^T x) = w_0 + w_k^T x$, g función identidad

Estas funciones ($f_k(x)$) son miembros de una clase de métodos llamados funciones discriminantes para cada clase.

Consideremos el proceso de clasificación en términos del conjunto de funciones discriminantes lineales en x, $f_k(x) = w_k^T x + w_{k0}$ definidas en las k-clases convencionalmente etiquetadas por 1,2,.., K.

En forma similar que el clasificador de bayes, el clasificador basado en las k funciones discriminantes $f_k(x) : \Re^P \rightarrow \Re$ es

$G(x) = \arg\max_{k \in C} f_k(x)$ $\quad G(x) : \Re^P \rightarrow$ C={1,2,.., K}.

El efecto del clasificador es dividir $\Re^P$ en K regiones

$$R_k = \left\{ x / x \in R^p \quad y \quad G(x) = \arg\max_{k \in C} f_k(x) \right\}$$

La frontera de decisión entre las clases k y l está dada por el conjunto de puntos para los cuales $f_k(x) = f_l(x)$ esto es:

$$\{x : (w_{k0} - w_{l0}) + (w_k - w_l)^T x = 0\}, \qquad \text{(A.1)}$$

desde que la relación (A.1) es verdadera para un conjunto de observaciones de algún par de clases, el espacio de entrada es dividida en regiones constantes de clasificación mediante hiperplanos, como se ilustra en la figura A.1.

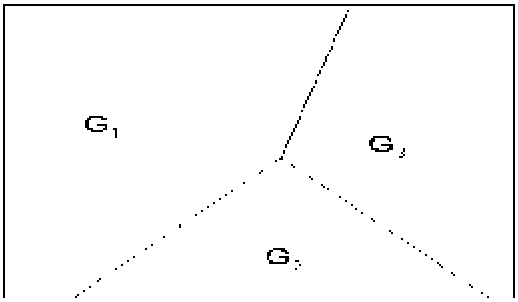


Figura A.1. Separación lineal.

En el caso particular de dos clases, K=2, la regla de clasificación es:

$$G(x) = \begin{cases} G_1 & si \quad f_1(x) > f_2(x) \\ G_2 & si \quad f_1(x) < f_2(x) \end{cases}$$

En forma equivalente

$$G(x) = f_1(x) - f_2(x)$$

En este caso la regla de clasificación es:

$$G(x) = \begin{cases} G_1 & si \quad G(x) > 0 \\ G_2 & si \quad G(x) < 0 \end{cases}$$

En forma general un vector de observaciones x, es asignado en la clase G_k si $f_k(x) > f_j(x) \quad \forall j \neq k$ Es decir $G(x) = k$

Estas formas de discriminante lineal pueden ser analizadas como formas de redes neuronales simples de adaptación de pesos entre las entradas y las salidas, presentaremos algunos clasificadores clásicos como perceptrón.

A2: CLASIFICADOR LINEAL COMO PERCEPTRON .

A.2.1 Clasificador gaussiano como perceptrón.

Sea $x \varepsilon \Re^p$ un vector de observaciones con vector de medias y matriz de covarianza definidos respectivamente por:

$$m = E(x)$$

$$C = E[(x-m)(x-m)^{'}]$$

Asumiendo que el vector x tiene distribución Gaussiana con función de densidad de probabilidad:

$$h(x) = \frac{1}{(2p)^{p/2}\left|C\right|^{1/2}} e^{-\frac{1}{2}(x-u)^T C^{-1}(x-u)}, \qquad \text{(A.2)}$$

y que se tienen dos grupos, entonces la función de densidad del vector x es:

$$h_k(x/G_k) = \frac{1}{(2p)^{p/2}\left|C\right|^{1/2}} e^{-\frac{1}{2}(x-u_k)^T C^{-1}(x-u_k)} \qquad \text{k=1,2} \qquad \text{(A.3)}$$

Donde:

m_k : vector de medias del i-esimo grupo.

C: matriz de covarianza de los dos grupos.

Tomando logaritmos en (A.2) se tiene:

$$\ln h_k(x/G_k) = -\frac{p}{2}\ln(2p) - \frac{1}{2}\ln\left|C\right| - \frac{1}{2}(x-u_K)^T C^{-1}(x-u_K) \qquad \text{k=1,2}$$

$$l_k = \ln h_k(x/G_k) = -\frac{p}{2}\ln(2p) - \frac{1}{2}\ln\left|C\right| - \frac{1}{2}x^T C^{-1}x + m_k^T C^{-1}x - \frac{1}{2}m_k^T C^{-1}m_k ,$$

el logaritmo de la función de máxima verosimilitud para el grupo G_1 esta dado por:

$$l_1 = \ln h_1(x/G_1) = -\frac{p}{2}\ln(2p) - \frac{1}{2}\ln|C| - \frac{1}{2}x^T C^{-1}x + m_1^T C^{-1}x - \frac{1}{2}m_1^T C^{-1}m_1$$

y para el grupo G_2 dado por:

$$l_2 = \ln h_2(x/G_2) = -\frac{p}{2}\ln(2p) - \frac{1}{2}\ln|C| - \frac{1}{2}x^T C^{-1}x + m_2^T C^{-1}x - \frac{1}{2}m_2^T C^{-1}m_2 \ ,$$

restando l_1 , l_2 y simplificando se tiene:

$$l_1 - l_2 = -\frac{1}{2}x^T C^{-1}x + \frac{1}{2}m_1^T C^{-1}x + \frac{1}{2}x^T C^{-1}m_1 - \frac{1}{2}m_1^T C^{-1}m_1 + \frac{1}{2}x^T C^{-1}x - \frac{1}{2}m_2^T C^{-1}x -$$

$$\frac{1}{2}x^T C^{-1}m_2 - \frac{1}{2}m_2^T C^{-1}m_2 \qquad \text{(A.4)}$$

como

$$m_1^T C^{-1}x = x^T C^{-1}m_1$$

$$m_1^T C^{-1}m_2 = m_2^{T'} C^{-1}m_1 ,$$

reemplazando estos últimos resultados se tiene

$$l_1 - l_2 = x^T C^{-1}(m_1 - m_2) - \frac{1}{2}(m_1 + m_2)^T C^{-1}(m_1 - m_2)$$

$$l_1 - l_2 = (m_1 - m_2)C^{-1}x^T - \frac{1}{2}(m_1 + m_2)^T C^{-1}(m_1 - m_2)$$

La estimación de máxima verosimilitud de los parámetro(pesos) w, está definida por :

$$l_1 - l_2 = w^T x - w_0 \qquad \text{(A.5)}$$

Donde $w = \Sigma^{-1}(m_1 - m_2)$ y $w_0 = \frac{1}{2}(m_1 + m_2)'\Sigma^{-1}(m_1 - m_2)$

La ecuación (A.5) representa en términos de redes neuronales la salida $g(Net) = g\left(w^T x - w_0\right) = w^T x - w_0$ con g función identidad. Por lo tanto el

clasificador máxima verosimilitud gaussiano puede ser implementado como perceptron mediante una combinación lineal tal como se observa en la figura.

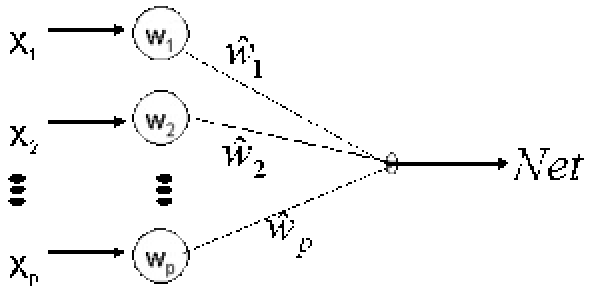


Figura A.2. Clasificador Gaussiano.

Si *Net* 0 ($l_1 \geq l_2$), se asigna x a G_1

Si *Net*<0 ($l_1 < l_2$), se asigna x a G_2

El clasificador gaussiano está relacionado con el perceptrón en que ambos son clasificadores lineales; sin embargo, existe alguna diferencia entre ellos, en el momento de la estimación de los parámetros. El perceptron no asume ninguna suposición a cerca de la distribución de los datos, en cambio el clasificador gaussiano asume normalidad para la distribución.

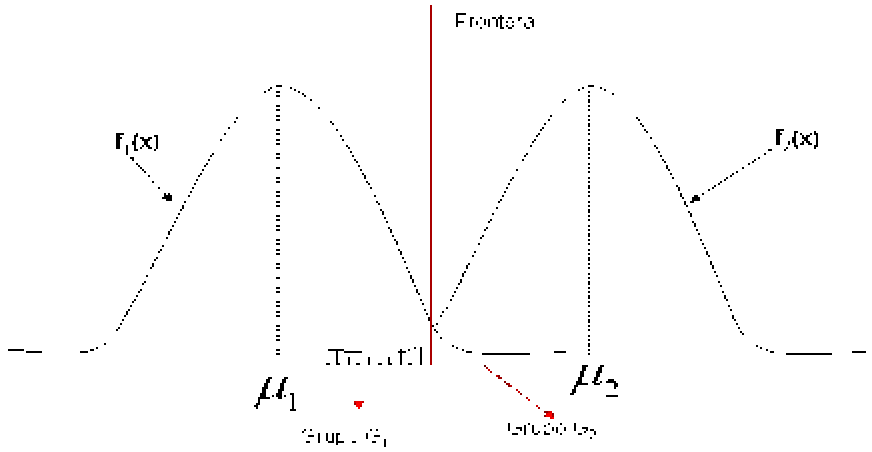


Figura A.3: Superposición entre dos distribuciones gaussianas.

A.2.2 Clasificador logístico como perceptrón.

Anteriormente hemos considerado funciones discriminantes como simples funciones lineales de las variables de entrada, hay varias formas en las cuales tales funciones pueden ser generalizadas. Se considera en este desarrollo el uso de una función no lineal g(.) que actúa sobre la Net para dar una función discriminante de la forma.

$$f(x) = g(Net) \tag{A.6}$$

Como una motivación de estas formas de funciones discriminantes, consideremos un problema de dos clases en el cual la densidad conjunta condicional en las clases está dada por distribuciones gaussianas (A.4) con matrices de covarianza iguales. Usando teorema de Bayes, la probabilidad posterior para los miembros de la clase G_1 es dada por:

$$P(G = G_1 / X = x) = \frac{h_1(x/G_1)\boldsymbol{p}_1}{h_1(x/G_1)\boldsymbol{p}_1 + h_2(x/G_2)\boldsymbol{p}_2} = \frac{1}{1+\left(h_2(x/G_2)\boldsymbol{p}_2 / h_1(x/G_1)\boldsymbol{p}_1\right)}$$

$$P(G = G_1 / \ X = x) = \frac{1}{1+\exp\left(-\ln\left(h_1(x/G_1)p_1 / h_2(x/G_2)p_2\right)\right)} = g(Net) \tag{A.7}$$

para este caso la función de activación g(Net) es la función exponencial(logística) y la Net está dada por:

$$Net_k = -\ln\left(h_1(x)p_1 / h_2(x)p_2\right) = w_k^T x + w_0$$

$$w = \Sigma^{-1}(m_1 - m_2) \quad \text{y} \quad w_0 = -\frac{1}{2}m_1^T \Sigma^{-1} m_1 + \frac{1}{2}m_2^T \Sigma^{-1} m_2 + \ln\frac{p_1}{p_2}$$

En forma similar se puede obtener las probabilidades posteriores para los miembros de la segunda clase G_2.

El uso de la función de activación *logística* en redes neuronales es muy importante, puesto que las salidas de la red están dadas por probabilidades posteriores.

Relacionando las ecuaciones (A.7) y (A.6) se tiene

$$f_k(x) = P(G = k / x) \qquad \text{(A.8)}$$

El criterio de clasificación es el siguiente:

Si $P(G = G_1 / X = x) > P(G = G_2 / X = x)$ se asigna x a la clase 1, en caso contrario a la clase 2. En resumen El clasificador esta dado por:

$$G(x) = \arg\max_{k \in C} f_k(x) \qquad \text{(A.9)}$$

Observaciones.

Si $|Net|$ es pequeño, entonces la función logística g(.) puede aproximar una función identidad, en este sentido una red con funciones de activación identidad es un caso particular de redes con funciones de activación logística (exponencial).

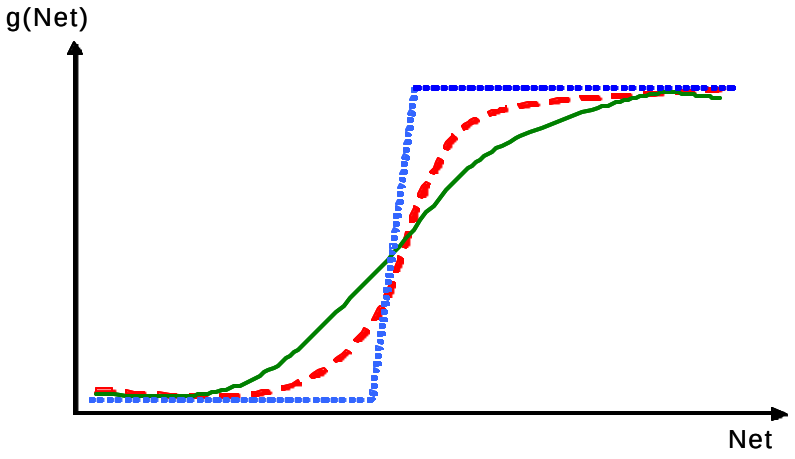


Figura A.4: Función de transferencia Logística

A.2.3 Interpretación geométrica del aprendizaje del perceptrón.

Consideremos el problema de clasificación de dos clases, las funciones discriminantes para cada una de las clases están dados por:

$$f_k(x) = w_{0k} + w_k^T x \quad k = 1,2 \ ,$$

y una regla de clasificación dado por:

$$G(x) = f_1(x) - f_2(x) \ ,$$

tal que el vector de observaciones x es asignado a la clase G_2 si G(x) <0 y a G_1 si G(x)>0. El clasificador

$$G(x) = w^T x + w_0$$

tiene por frontera (G(x)=0) un hiperplano de dimensión (p+1) y esta dado por:

$$L : \{x : w_0 + w^T x = 0\}. \qquad \text{(A.10)}$$

En el caso particular de p=2, la frontera está ilustrada en la figura 1.11. Si x_A y x_B son puntos del hiperplano, entonces, $f_1(x_A) = f_2(x_B) = 0$ de donde:

$$w^T (x_A - x_B) = 0 \quad \text{y} \quad w^* = w/\|w\| \qquad \text{(A.11)}$$

w es normal a algún vector en el hiperplano, específicamente determina la orientación de la frontera, por lo tanto la condición de correcta operación del perceptrón es que este hiperplano divida al conjunto de entrada en dos subconjuntos, de manera que cada uno agrupe las entradas asociadas a un mismo tipo de salida.

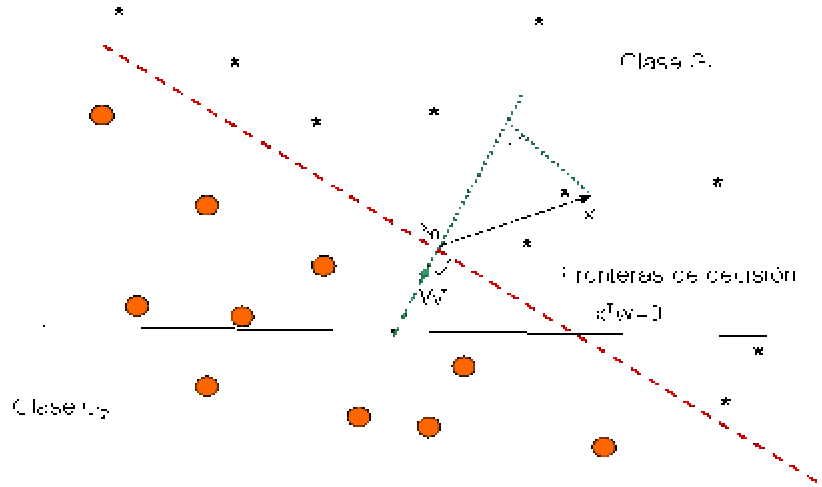


Figura A.5 Perceptrón como Clasificador Lineal.

A3: SESGO Y VARIANZA.

El análisis "sesgo varianza" es una herramienta fundamental para comprender el comportamiento de cualquier algoritmo de estimación. Dicho análisis resulta ser una técnica natural en análisis de regresión con pérdida cuadrática debido a las propiedades de ésta, en relación con el problema de clasificación con pérdida 0-1, se han propuesto diferentes descomposiciones del valor esperado del error de predicción tratando de emular la descomposición sesgo varianza del problema de regresión lineal.

A.3.1 Regresión Lineal.-

Si asumimos que $Y = f(X) + e$ donde $E(e) = 0$, $\text{var}(e) = s_e^2$ y $f(X)$ es el modelo real; esto es, $f(x) = E(Y / X = x)$.

En el caso de las redes neuronales, si $\hat{f}(X)$ estima $f(X)$, el valor esperado del error de predicción que se obtiene es:

$$Err(X) = E\left[L\left(Y, \hat{f}(X)\right)\right]$$

$$Err(x_0) = E\{(y - \hat{f}(x_0))^2\}$$

$$= E\{(f(X) + e - \hat{f}(x_0))^2\}$$

$$= E\{(e)^2\} + \{(f(x_0) - \hat{f}(x_O))^2\} + E\{2(e).(f(x_0) - \hat{f}(x_O))\}$$

Puesto que e, es independiente de $f(X)$ y $\hat{f}(X)$, la relación se reduce a:

$$Err(x_0) = E\{(y - f(x_0))^2\} + E\{(f(x_0) - \hat{f}(x_O))^2\} \qquad \text{(A.12)}$$

Desarrollando el segundo termino de (A.12)

$$E\{(f(x_0) - \hat{f}(x_O))^2\} = E\{(f(x_0) - E(\hat{f}(x_0)) + E(\hat{f}(x_0)) - \hat{f}(x_O))^2\}$$

$$= E\{(f(x_0) - E(\hat{f}(x_0)))^2\} + E\{(\hat{f}(x_O) - E(\hat{f}(x_0)))^2\}$$

$$+ E\{2.(f(x_0) - E(\hat{f}(x_0))).(\hat{f}(x_O) - E(\hat{f}(x_0)))\}$$

Como $E\{\hat{f}(x_O) - E(\hat{f}(x_0)\} = E(\hat{f}(x_0)) - E(\hat{f}(x_0)) = 0$

$$E\{(f(x_0) - \hat{f}(x_O))^2\} = \{(f(x_0) - E(\hat{f}(x_0)))^2\} + E\{(\hat{f}(x_O) - E(\hat{f}(x_0)))^2\}, \qquad \text{(A.13)}$$

de (A.12) y (A.13) tenemos la siguiente descomposición del valor esperado del error de predicción.

$$Err(x_0) = E\{(Y - f(x_0))^2\} + \{(f(x_0) - E(\hat{f}(x_0)))^2\} + E\{(\hat{f}(x_O) - E(\hat{f}(x_0)))^2\}$$

$$Err(x_0) = E\{(Y - f(x_0))^2\} + \{(E(\hat{f}(x_0)) - f(x_0))^2\} + E\{(\hat{f}(x_O) - E(\hat{f}(x_0)))^2\}$$

$$Err(x_0) = s_e^{\,2} + Sesgo^2(\hat{f}(x_O) + \mathrm{var}(\hat{f}(x_O)$$

$$Err(x_0) = Error\ \ irreducible + Sesgo^2 + \mathrm{var}ianza, \qquad \text{(A.14)}$$

donde:

s_e^2: el error residual, es el error obtenido por el modelo real, el cual es independiente del algoritmo de aprendizaje.

$Sesgo = E(\hat{f}(x_0)) - f(x_0)$: es una medida de discrepancia entre el modelo real y el modelo promedio.

$Varianza = E\{(\hat{f}(x_O) - E(\hat{f}(x_0)))^2\}$: es la medida de variabilidad de las predicciones con respecto al conjunto de datos de entrenamiento.

A.3.2 Clasificación.-

Konavi y Wolper (1996) propusieron una descomposición del valor esperado del error de predicción para la clasificación, usando pérdida 0-1, para ello definieron unas nuevas variables numéricas como:

$$\hat{f}^k(x_o) = I(\hat{G}(x_o) = G_K) \quad \text{y} \quad Y_k = I(G = G_k) = \begin{cases} 1 & si \quad G = k \\ 0 & si \quad G \neq k \end{cases}, \text{ k=1,..,K}$$

$$Err(x_0) = E\left[I\left(\hat{G}(x_0) \neq G\right)\right] = E\left[\frac{1}{2}\sum_{k=1}^{K} I\left(Y_k \neq \hat{f}^k(x_0)\right)\right]$$

$$= E\left[\frac{1}{2}\sum_{k=1}^{K}\left(Y_k - \hat{f}^k(x_0)\right)^2\right] = \frac{1}{2}\sum_{k=1}^{K} E\left(Y_k - \hat{f}^k(x_0)\right)^2,$$

en forma similar que la descomposición de regresión tenemos para un k:

$$E\left(Y_k - \hat{f}^k(x_0)\right)^2 = E\left\{\left(Y_k - E(Y_k)\right)^2\right\} + \left\{\left(E(\hat{f}^k(x_0)) - E(Y_k)\right)^2\right\} + E\left\{\left(\hat{f}^k(x_O) - E(\hat{f}^k(x_0))\right)^2\right\}$$

$$= \left(p(G_k / x_0) - \left(p(G_k / x_0)\right)^2\right) + \left(p(G_k / x_0) - p(G(x_0) = G_k)\right)^2$$

$$+ \left(p(G(x_0) = G_k) - \left(p(G(x_0) = G_k)\right)^2\right),$$

puesto que:

$E(Y_k) = E\left((Y_k)^2\right) = p(G_k / x_0)$, probabilidades iniciales.

$E\left(f^k(x_0)\right) = E\left(f^k(x_0)\right)^2 = p(G(x_0) = G_k)$, salida de la red que representan probabilidades posteriores.

Tenemos la descomposición del valor esperado del error de predicción:

$$Err(x_0) = s_e^{\,2}(x_0) + sesgo(x_0) + \text{var}\,ianza(x_0) \qquad \text{(A.15)}$$

$$s_{KW}^{\,2}(x_0) = \frac{1}{2}\left(1 - \sum_k \left(p(G_k / x_0)\right)^2\right)$$

$$sesgo_{KW}(x_0) = \frac{1}{2}\sum_k \left[p(G_K / x_0) - p(G(x_0) = G_k)\right]^2$$

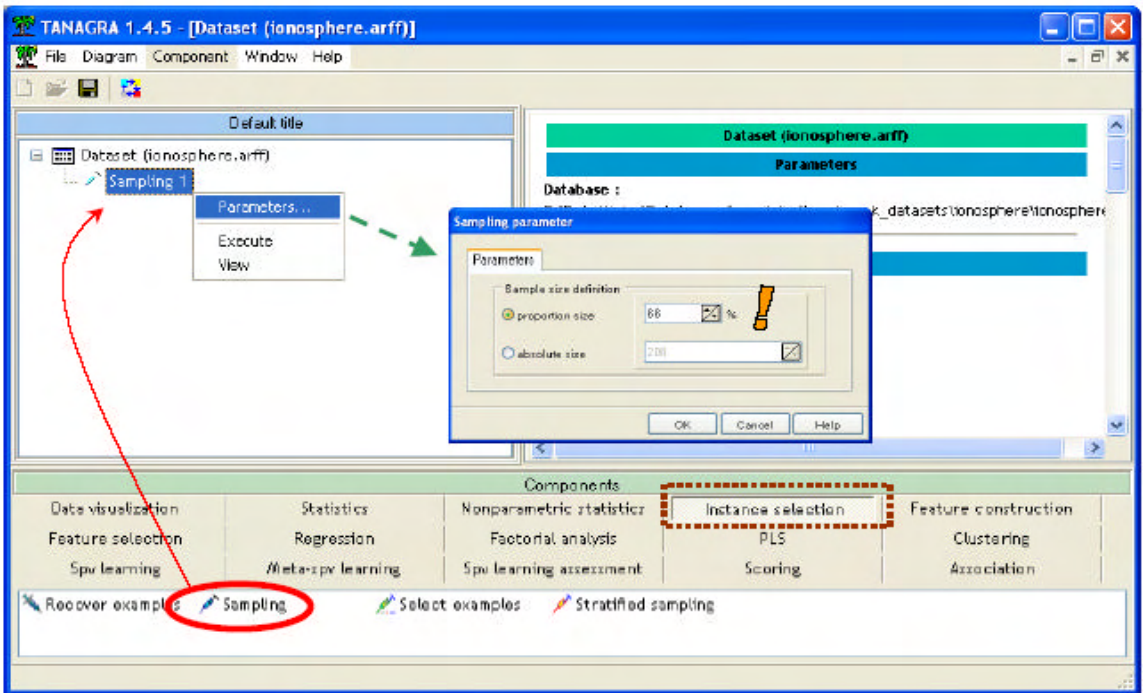
$$\text{var}\,ianza_{KW}(x_0)] = \frac{1}{2}\left(1 - \sum_k \left(p(G(x_0) = G_k)\right)^2\right)$$

APÉNDICE B

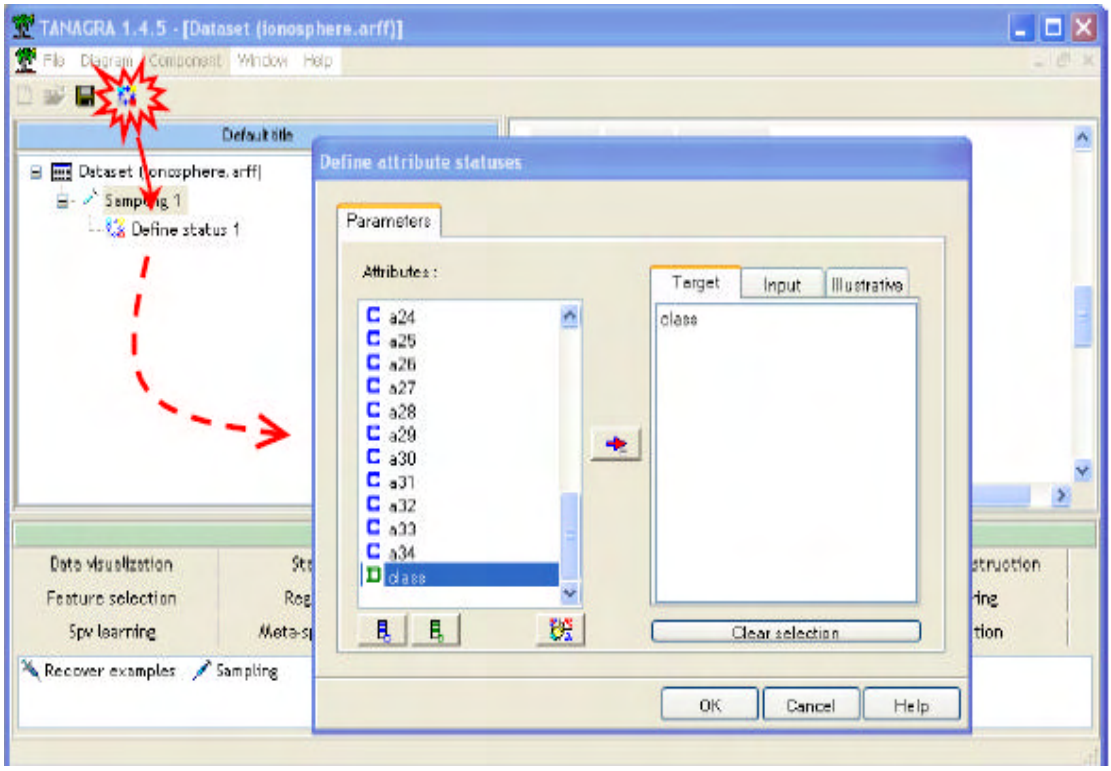
B1: Entrenamiento de la red neuronal con el programa tanagra.

Para entrenar la red neuronal con el programa Tanagra 1.4.12 se debe seguir los siguientes pasos:

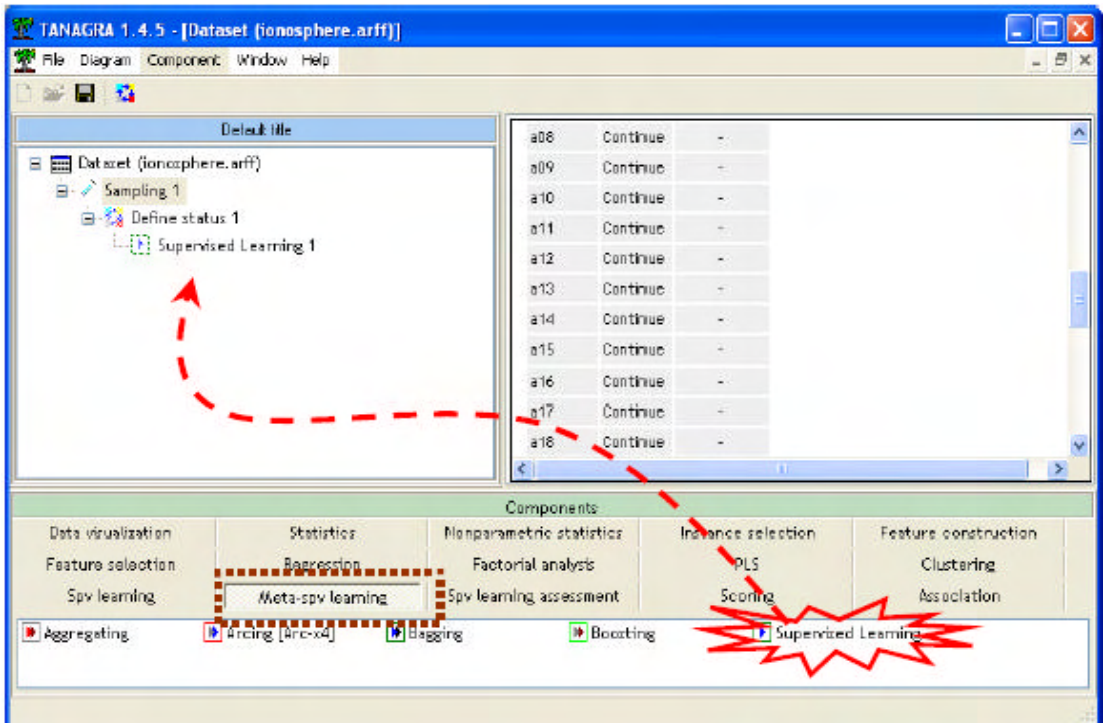
- Importar la base de datos.
  File → New → Dataset → (Nombre del archivo).
- En la barra de componentes activar *Instance selection* y seleccionar sampling, como se ilustra en el grafico siguiente.



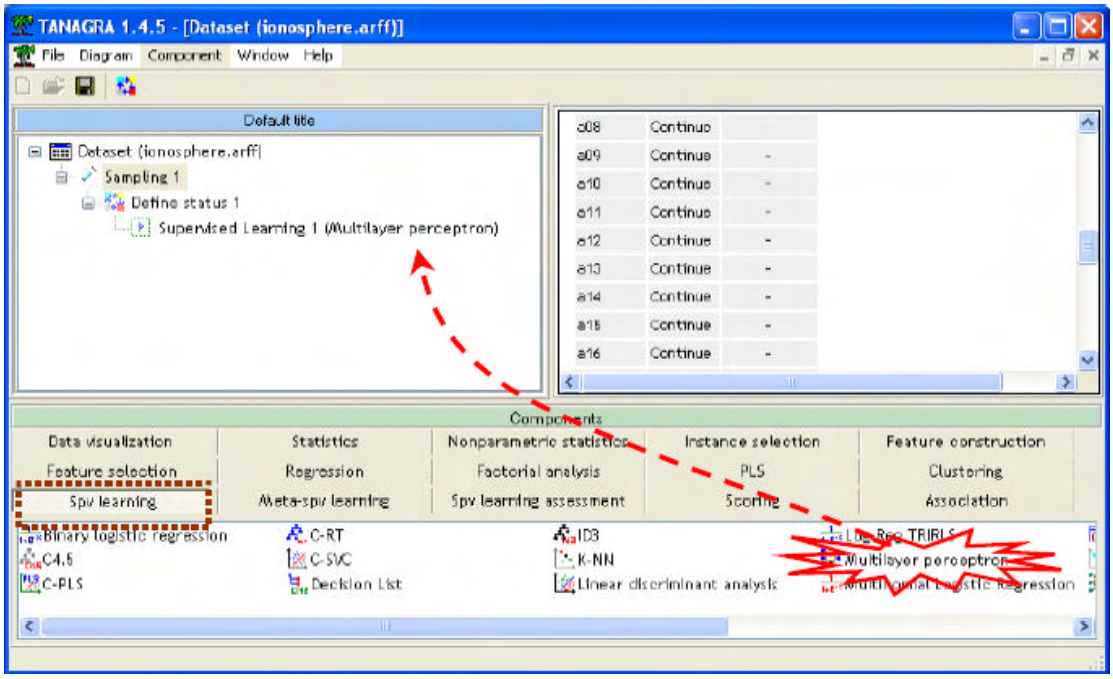
- Defina las variables independientes y dependientes.



- Seleccione en el menú de componentes, aprendizaje supervisado.



- Seleccione el método de aprendizaje.

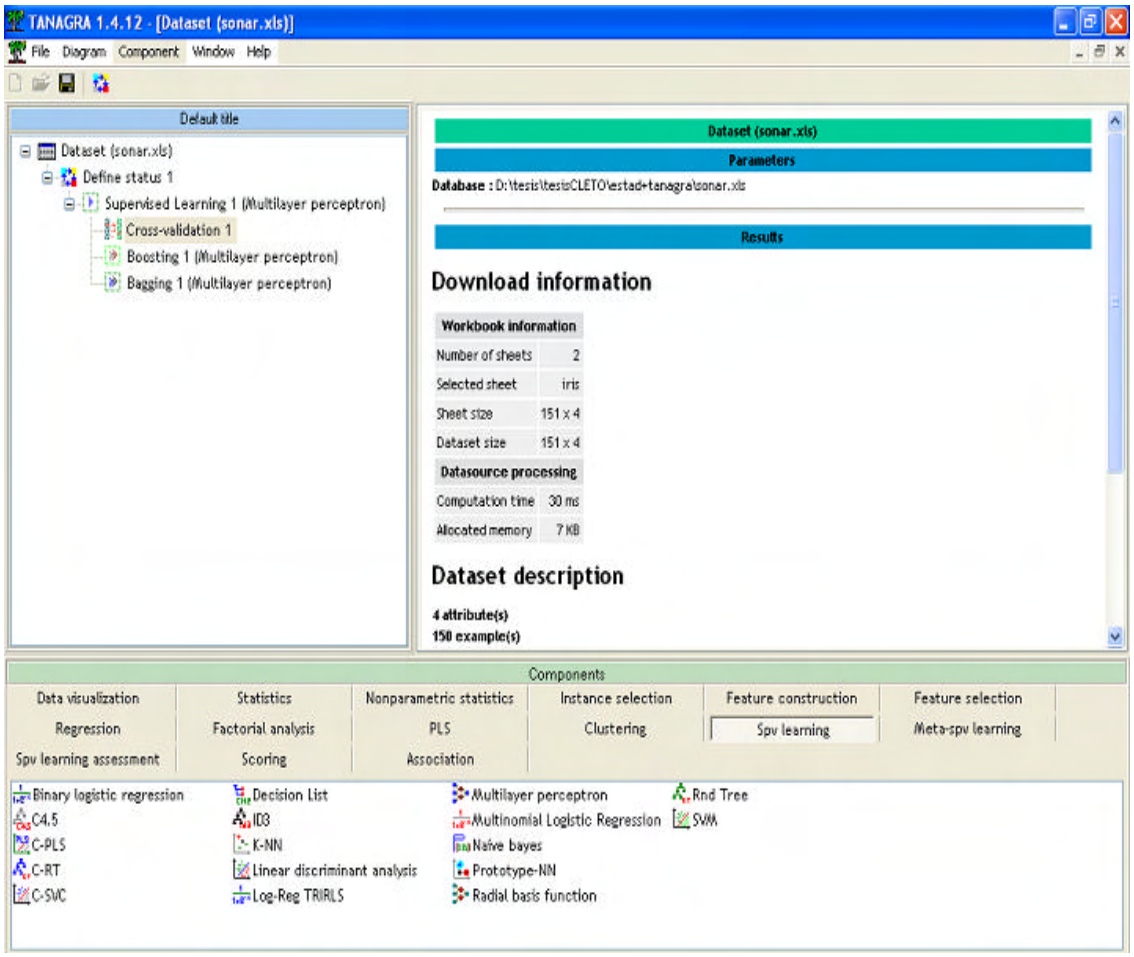


- Haga anticlic en supervised learning 1(multiplayer perceptron) para especificar las características de la red, esto es número de neuronas en la capa oculta, factor de aprendizaje y regla de parada.
- Finalmente para visualizar los resultados seleccione *View* del menú.

B2: Bagging y Boosting con el programa tanagra.

La secuencia para realizar Bagging y Boosting es la siguiente:

- Entre la red neuronal
- En el menú de componentes seleccione Spv learning assessment y elija cross-validation.
- En el menú de componentes seleccione Meta-spv learning y elija bagging y boosting.
- Finalmente especifique las propiedades de los métodos de combinación de clasificadores tales como numero de clasificadores base y las características de la red. Este proceso se ilustra en el grafico siguiente.



BIBLIOGRAFÍA

1. Acuña, E. and Daza, L. (2003). Combining classifiers based on gaussian mixtures, Univesity of Puerto Rico at Mayaguez.
2. Acuña, E. and Rojas , A. (2004). Ensembles of classifiers based on Kernel density estimators.
3. Bauer, Kohavi and Kluwer. (1999). An empirical comparison of voting clasification algorithms: bagging, boosting and variants, Academic publishers, Boston.
4. Bishop, C. (1997). Neural networks for pattern recognition-. Clarendon Press, Oxford.
5. Breiman, L. (1996). Bagging predictors, Department of Statistics Univerity of California.
6. Breiman, L. (1996). Bias , variance and arcing classifiers, Department of Statistics, Univerity of California,
7. Cheng and Titterington. (1994). Neural networks: a review from a statistical perspective, Statistical Science.
8. Friedman, J., Hastie, T. and Tibshirani, R. (1999). Additive logistic regression a statistical view of boosting, Annals of Statistics **28**.
9. Friedman, J., Hastie, T., Tibshirani, R. (2001). The elements of statistical learning data mining, inference and prediction, Springer.
10. Freud and Schapire. (1996). Boosting , the margin: a new explanation for the effectiveness of voting methods, USA.
11. Hastie, T. and Tibshirani, R. (1990). Generalized additive models, Chapman & Hall.
12. Haykin, S. (1994). Neural networks, Macmillan College Publishing Company
13. Hilera, J. y Martínez, J. (1995). Redes neurales artificiales- Addison-Wesley Iberoamericana.
14. Kohavi and Wolpert . (1996). Bias plus variance decomposition for zero-one loss function, Data Mining and Visualization.

15. Krogh and Vedelsby. (1995). Neural networks ensembles, cross validation, and active learning, Advancs in Neural Information, Cambridge.
16. Optiz and Maclin. (1999). Popular ensemble methods: an empirical study,Journal of Artificial Intelligence Research.
17. Schapire , R. and Singer , Y. (1998). Improved boosting algorithms using confidence rated predictions proceeding of the eleventh annuals conference on computational learning theory.
18. Tibshirani, R. (1996). Bias, variance and prediction error for clasification rules, University Toronto.

Lima Mayo 2007